UNIVERSIDAD CARLOS III DE MADRID

DEPARTAMENTO DE INFORMÁTICA

INGENIERÍA INFORMÁTICA

PROYECTO FIN DE CARRERA

Requirements Studio Plus

Una herramienta para la organización y control de calidad de los requisitos del software

Autor: Alexandre Vázquez Vázquez

Director: Gonzalo Génova Fuster

Leganés, octubre de 2010

Índice de Contenidos

Índice de Ilustraciones

Índice de Tablas

1 Introducción

El objetivo del Proyecto Fin de Carrera que se expone en el presente documento es el de integrar y mejorar las funcionalidades y servicios ofrecidos por dos Proyectos Fin de Carrera anteriores desarrollados en el seno de la Universidad Carlos III de Madrid, denominados MeCaReq [Fernandez 08] y Requirements Studio [Alonso 08] a través de un proyecto nuevo que no repita errores de sus antecesores.

Este proyecto busca inicialmente la corrección y mejora de la herramienta MeCaReq con el fin de superar las limitaciones detectadas tras un uso intensivo de la misma. Posteriormente, se buscó la integración de las funcionalidades de dicha aplicación con la herramienta Requirements Studio, lo que hizo conveniente rehacer también esta segunda herramienta y tratar de mejorar algunos aspectos tanto de implementación como de funcionalidades, aunque en este caso conservando en mayor medida su concepción original.

Este Proyecto Fin de Carrera hace referencia a la organización de requisitos así como al estudio de las métricas de evaluación de calidad en los mismos. La ingeniería de requisitos es la rama de la ingeniería del software que se ocupa de una de las primeras etapas en el proceso de desarrollo del software: la comprensión de las necesidades del cliente y su definición en forma de conjunto estructurado de requisitos que debe satisfacer un sistema informático [Genova 08b].

A día de hoy los proyectos siguen siendo enormemente inestables, de forma que la mayoría de ellos se abandonan o no se completan con éxito [Domínguez 09], es decir, en el tiempo estimado, con el presupuesto y capacidades inicialmente acordadas tal y como se puede ver en la Tabla 1:

	1994	**1996**	**1998**	**2000**	**2002**	**2004**	**2006**	**2009**
Exitosos	16%	27%	26%	28%	34%	29%	35%	32%
Comprometidos	53%	33%	46%	49%	51%	53%	46%	44%
Abandonados	31%	40%	28%	23%	15%	18%	19%	24%

Tabla 1: Datos de proyectos exitosos en el *The Chaos Report*

Como ya se ha señalado en gran cantidad de fuentes, la mayor parte de los defectos en el software entregado tienen su origen en un deficiente análisis de requisitos, y son, en general, los más difíciles y costosos de reparar. Por eso es una prioridad dotar a la ingeniería de requisitos de disciplina ingenieril, en particular, mediante la realización de controles de calidad desde el inicio del proceso de construcción de un producto software. Si no se exige que los requisitos cumplan determinados criterios de calidad, entonces será más difícil buscar la calidad en fases posteriores del proyecto.

Cabe destacar que toda medición de calidad tiene un coste añadido en tiempo y dinero, por lo que la automatización de las métricas de calidad puede suponer un ahorro considerable frente a las tediosas evaluaciones realizadas de modo manual. Ahora bien, es importante resaltar que el objetivo final debe ser medir para mejorar. Reducir la gestión de la calidad a la obtención de una valoración numérica tropezaría con la oposición frontal de los propios ingenieros de requisitos, que no verían en las métricas la ayuda de un consejero, sino un "mecanismo policial de penalización". Para evitar esto, es necesario que los indicadores de calidad no se limiten a proporcionar valoraciones numéricas, sino sobre todo que señalen defectos concretos y proporcionen sugerencias para mejorarlos, de modo análogo a la forma en que un corrector ortográfico y gramatical puede ayudar a mejorar la calidad de un texto.

1.1 Motivación

En el momento actual, la gran mayoría de las empresas se encuentran realizando documentos de requisitos con grandes deficiencias y que generan numerosas pérdidas de tiempo, dinero y confianza de sus clientes. La calidad en un requisito es muy difícil de medir, debido a que la mayoría de las propiedades que determinan la calidad son "subjetivas" o difícilmente evaluables. El objetivo es prestar una ayuda a los escritores de requisitos para que cometan los menores errores posibles. Para ello, es necesario disponer de una métrica que permita determinar qué requisitos tienen calidad y cuáles no la tienen.

Otro de los problemas existentes es como generar dichas métricas, partiendo de que existen diferentes formas de pensar respecto a este hecho. Es conocido que existen responsables de proyecto o departamentos de calidad que les gusta tener un control exhaustivo sobre el proceso de calidad de los requisitos, con lo cual es lógico que un usuario pueda determinar qué propiedades serán tenidas en cuenta y qué condiciones deben cumplirse para que un requisito sea calificado como bueno o como malo. Sin embargo, también es posible, que las decisiones que tome el responsable no sean las más adecuadas o puedan ser mejoradas a partir de un proceso más objetivo, y ahí entra la necesidad de disponer de un proceso automático que permita obtener una métrica de calidad, más allá que la pura intuición o de la arbitrariedad. Para ello, la forma más adecuada es la extracción de patrones o de características comunes en requisitos que han sido evaluados por parte de un evaluador humano y que han sido calificados como bueno. Es decir, se pretende dar respuesta ¿qué propiedades tiene este requisito que hace que el ser humano que lo ha evaluado lo califique como bueno?

Según lo anteriormente expuesto, se observa que existe la necesidad de aplicaciones que se encarguen de obtener una valoración automática que faciliten la tarea de los encargados de redacción de requisitos. Del mismo modo, se puede intuir que la obtención de métricas que se encarguen de la evaluación de requisitos es una tarea complicada ya que no existe un conjunto de normas y pautas a seguir.

Esto es debido, principalmente, a que las propiedades calidad que se exigen normalmente en la redacción de un requisito, tales como claridad, desambigüedad, verificabilidad, son difícilmente medibles por procesos puramente automáticos. Con lo cual, es necesario realizar un trabajo previo de adaptar y asociar esas características que se quieren medir a otras características que sí puedan ser medidas realmente de un modo objetivo.

Sin embargo, esa adaptación o correspondencia no es ni mucho menos perfecta, con lo que es necesario hacer uso de algún tipo de *feedback* o de valoración por parte de un "evaluador humano" que permita extraer unas directrices de qué características comunes tienen aquellos requisitos que son calificados como buenos. Es decir, la labor de obtención de una métrica consiste en extraer patrones comunes sobre un conjunto de requisitos que un evaluador humano ha calificado como "buenos" y qué es lo que los diferencia de otros que han sido calificados como "malos". De este modo, facilita la obtención de una métrica compuesta por un conjunto de características que indican que "diferencias" existen entre los requisitos que han sido redactados y qué correcciones deben realizarse hacer para que se 'asemejen' a aquellos propios del conjunto de entrenamiento que han sido calificado como "buenos".

Del mismo modo, con el fin de que esta herramienta sea realmente útil para los usuarios, su uso deber fácil y debe de poder utilizarse como un instrumento básico de trabajo, similar a lo que ocurre con el corrector ortográfico que se encuentra incrustado en el procesador de textos, con lo que la herramienta debe tener una integración máxima con un organizador de requisitos de forma que el trabajo de evaluación sea más cómodo.

1.2 Visión general del documento

Este documento, en el cual se presenta el Proyecto Fin de Carrera que se encarga tanto de la gestión y organización de requisitos así como el estudio y generación de métricas de calidad en requisitos, se compone de los siguientes capítulos:

- Un primer capítulo de introducción, en el cual se describen brevemente los objetivos y motivaciones que se pretenden conseguir mediante la realización de este proyecto.

- En un segundo capítulo se documentan las bases teóricas en las cuales se asienta este proyecto con el fin de que el lector pueda tener unas nociones básicas de aquellos conceptos utilizados a lo largo del presente documento.

- Posteriormente, en el tercer capítulo, se realiza la descripción básica del proyecto, a través de una descripción general, característica de los usuarios potenciales del sistema, características y restricciones generales, así como suposiciones y dependencias.

- En el cuarto capítulo, se realiza una descripción más detallada de las características de la aplicación, mediante la formalización de los requisitos software de la aplicación, así como del modelo conceptual del mismo.

- En el quinto capítulo, se presenta el primer paso del diseño, mediante la descomposición arquitectónica del mismo y la descripción básica de los componentes que conforman la solución implementada.

- En el sexto capítulo, se realiza la descripción de aquellas partes más importantes a tener en cuenta en el proceso de implementación del sistema con el fin de facilitar el mantenimiento del mismo.

- En el séptimo capítulo, se describe el conjunto de pruebas que se han realizado sobre la aplicación, con el fin de asegurar un cierto nivel de fiabilidad y confianza en el software generado.

- En el octavo capítulo, se muestran datos asociados al proceso de realización del sistema, como pueden ser la planificación o el presupuesto del mismo.

- En el noveno capítulo, se realizan unas conclusiones sobre el desarrollo del proyecto, donde se indican tanto la situación actual del proyecto, así como las dificultades que han sido detectadas y sorteadas durante el proceso de realización del proyecto, así como posibles líneas de desarrollos futuros con el fin de complementar el ámbito de este proyecto.

.

2 Estado del Arte

2.1 Métricas de Calidad en Requisitos

Una de las bases teóricas donde se asienta gran parte de la realización de este proyecto, radica en la obtención de métricas que permitan determinar con una cierta confianza de manera automática la calidad de un determinado requisito. La gran parte de la base teórica se extrae del artículo [Genova 08c] y se basa principalmente en determinar cuáles son las propiedades de las descripciones de los requisitos que permiten determinar su nivel de calidad. Muchos autores y metodologías distinguen entre propiedades cualitativas deseables de los requisitos, que dependerían del juicio subjetivo, e indicadores cuantitativos medibles, basados en características objetivas de los requisitos [Rosenberg 01] y [Wilson 97].

En la documentación relacionada existen diferentes listas de propiedades deseables que determinan la calidad de un requisito, pero todas las características son complementarias: completitud, corrección, jerarquización, inambigüedad, consistencia, modificabilidad, trazabilidad, verificabilidad, validabilidad, comprobabilidad [IEEE 830]; completitud, consistencia, claridad, comprobabilidad, existencia de condiciones de error, trazabilidad [Braude 01]; realismo (o viabilidad), verificabilidad, comprensibilidad, trazabilidad, adaptabilidad [Sommerville 04]; viabilidad, claridad, trazabilidad, comprobabilidad, consistencia, abstracción, inambigüedad, cuantificación, estructuración, existencia de fuente [Pressman 05]; inambigüedad, completitud, consistencia, comprensibilidad [Fabrini 01].

Cabe destacar que los indicadores no son lo mismo que las propiedades, pero están más o menos directamente relacionados con ellas, de forma que si los indicadores cuantitativos mejoran, las propiedades cualitativas afectadas por ellos mejorarán, de modo que los indicadores pueden utilizarse como medida objetiva de la calidad subjetiva.

Es necesario justificar la lista de propiedades deseables que se espera que cumpla una especificación de requisitos, a partir de la siguiente pregunta: ¿Cuál es el objetivo de los requisitos? La respuesta indicará qué se debe considerar como requisitos “buenos” y “malos”. Pues bien, el objetivo de los requisitos es especificar un sistema informático: el sistema que el cliente necesita, y que el ingeniero produce y mantiene.

Desde el punto de vista del cliente, por tanto, la cualidad final de los requisitos (en su conjunto e individualmente) es la validabilidad, es decir, que el cliente sea capaz de

confirmar que los requisitos expresan efectivamente el sistema que responde a sus necesidades. Esta propiedad se puede desglosar, en un segundo nivel, en otras tres: completitud (están cubiertas todas las necesidades), consistencia (no hay contradicciones entre unos requisitos y otros), y comprensibilidad (los requisitos se entienden correctamente sin dificultad).

Desde el punto de vista del ingeniero de requisitos, en cambio, se puede considerar que las dos cualidades esenciales son: verificabilidad, es decir, es posible comprobar que el sistema producido se corresponde con el sistema especificado; y modificabilidad, ya que, si no es fácil modificar los requisitos, tampoco será fácil modificar el sistema producido durante la fase de mantenimiento del ciclo de vida. La verificabilidad depende de las mismas tres propiedades "de segundo orden" que se han mencionado con anterioridad (completitud, consistencia y comprensibilidad), así como de otras tres: inambigüedad (existencia de una interpretación unívoca para cada requisito), trazabilidad (existencia de una relación explícita de cada requisito con artefactos de diseño, implementación y prueba) y abstracción (decir qué debe hacer una aplicación sin decir cómo debe hacerlo, es decir, evitar el exceso de detalle en la forma de especificar el requisito); de estas dos últimas también depende estrechamente la modificabilidad.

La inambigüedad y la comprensibilidad están relacionadas entre sí (según algunos, serían la misma propiedad), ya que si un requisito es ambiguo no puede ser propiamente comprendido. Es posible distinguir finalmente un tercer nivel más básico en esta taxonomía de propiedades deseables, que contendría otras dos propiedades aún más elementales: precisión (ausencia de términos vagos o mal definidos) y atomicidad (determinación e identificación clara de cada requisito, sin mezclarlo con otros requisitos). De la precisión dependen la completitud, la consistencia, la comprensibilidad y la inambigüedad; de la atomicidad dependen la completitud, la precisión, la abstracción (un requisito no atómico corre el riesgo de estar excesivamente detallado), y muy especialmente la trazabilidad, y por tanto la modificabilidad en la fase de mantenimiento del ciclo de vida. Efectivamente, si los requisitos constituyen un sistema monolítico, en lugar de un sistema estructurado en elementos individuales bien identificados y relacionados entre sí, entonces no será fácil ni modificar los requisitos ni modificar el sistema producido. La Ilustración 1 presenta esta taxonomía de propiedades que acaban de ser explicadas:

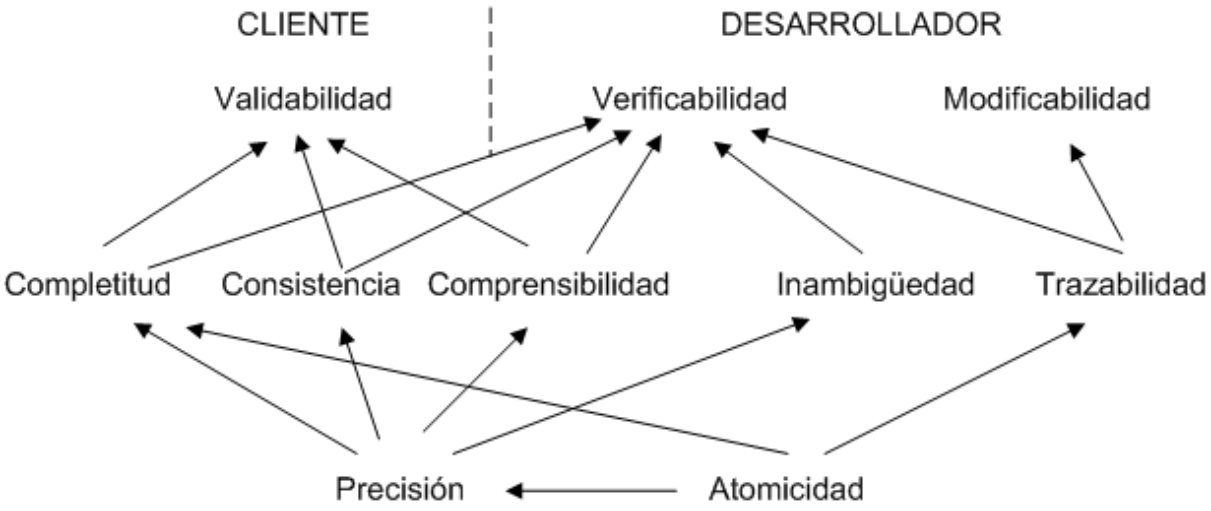

Ilustración 1: Influencia de unas propiedades cualitativas en otras

2.1.1 Indicadores medibles en los requisitos

Una vez establecidas las características que deben ser medidas, es necesario establecer cómo medir esas propiedades deseables que han sido identificadas. Ya se ha mencionado que estas propiedades dependen del juicio subjetivo, lo que no significa que sean arbitrarias, sino que son cualidades difícilmente cuantificables. Así pues, es necesario definir una serie de indicadores cuantificables que estén relacionados con las propiedades cualitativas que se desean evaluar.

Por ejemplo, es posible usar como indicador el tamaño de un requisito, medido por el número de palabras de su descripción. El tamaño influye en las propiedades del requisito, particularmente en la atomicidad, y a través de ella en todas las demás. ¿Qué tamaño indica una buena atomicidad? El tamaño, como medida numérica directa de una característica del texto, no indica nada por sí mismo: obviamente, un requisito no es mejor por el simple hecho de ser más grande o más pequeño. Por este motivo, se suelen clasificar las mediciones en un conjunto de niveles discretos, normalmente tres o un número similar: Alto, Medio, Bajo; Bueno, Medio, Malo; etc.

En el caso del tamaño del requisito, parece claro que un buen requisito no debe ser ni muy pequeño ni muy grande. En otras palabras, para transformar una medida primitiva tal como el tamaño en valor numérico en un indicador de tamaño adecuado es necesaria una función escalonada convexa (es decir, en forma de "montaña", creciente-decreciente), como la de la Ilustración 2.

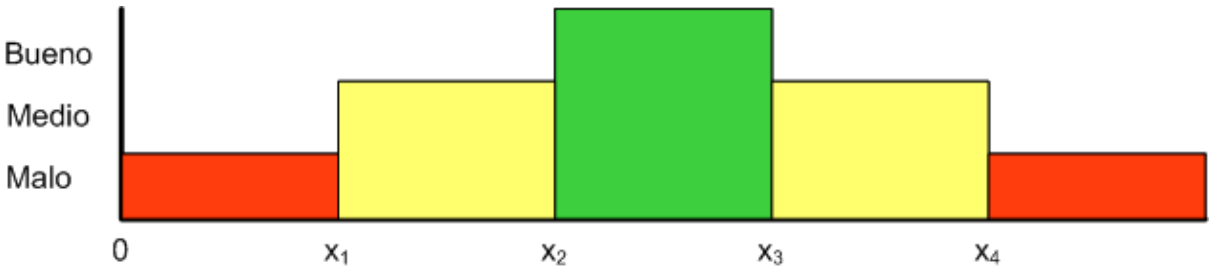

Ilustración 2: El indicador de "tamaño adecuado" en función del "tamaño numérico"

El tamaño se considera "bueno" si está entre los límites x_2 y x_3, "malo" si es inferior a x_1 o superior a x_4, y "medio" en los intervalos x_1-x_2 y x_3-x_4. La dificultad consiste, precisamente, en determinar el valor de los cuatro parámetros de esta función (los extremos de los intervalos).

Del mismo modo que con el tamaño, es posible proceder con otros indicadores tales como índice de legibilidad, número de formas imperativas, número de términos ambiguos, número de términos del dominio, etc.

Cada uno de estos indicadores tiene características distintas, de modo que la función escalonada de transformación será distinta para cada uno. Es decir, para algunos indicadores, como es el caso del tamaño, se considera adecuado un valor intermedio de la medida numérica primitiva, mientras que para otros lo adecuado es más bien un valor extremo. Por ejemplo, en el caso del número de términos ambiguos lo adecuado es que no haya ninguno, y cuantos más haya peor será el resultado del indicador: sería una función escalonada decreciente.

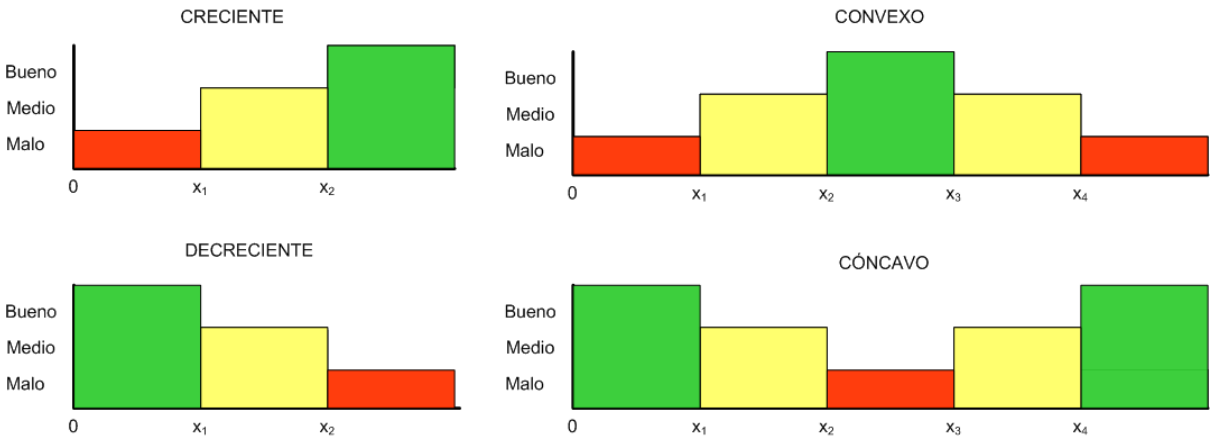

Ilustración 3: Tipos básicos de funciones escalonadas para transformar medidas numéricas a indicadores de calidad

En general, es posible distinguir cuatro tipos básicos de funciones escalonadas: creciente, decreciente, convexa y cóncava, que son las que muestran en la Ilustración 3. Las dos primeras requieren dos parámetros para quedar definidas (x_1, x_2), mientras que las dos últimas ("montaña" creciente-decreciente, y "valle" decreciente-creciente) requieren cuatro parámetros (x_1, x_2, x_3, x_4). Para evitar ambigüedades, se definen los intervalos cerrados en el extremo inferior y abiertos en el superior, es decir, $[0, x_1)$, $[x_1, x_2)$, $[x_2, \infty)$, o bien $[0, x_1)$, $[x_1, x_2)$, $[x_2, x_3)$, $[x_3, x_4)$, $[x_4, \infty)$.

Así, la función creciente da un valor Medio si el recuento numérico x cumple la doble condición $(x \geq x_1)$ y $(x < x_2)$, y análogamente en el resto de los casos. Si en algún caso se desea distinguir sólo dos niveles de calidad (Bueno, Malo), entonces bastará con igualar los parámetros $x_1 = x_2$ y $x_3 = x_4$ en la definición de la función.

El uso de funciones escalonadas, además de servir para transformar una medida numérica en un indicador de calidad, es útil también para uniformar medidas numéricas que por su propia naturaleza serán muy heterogéneas. Así, por ejemplo, el rango numérico del

tamaño de un requisito medido en palabras puede estar entre unas pocas unidades y unos pocos centenares, mientras que el rango de términos ambiguos siempre sería de unas pocas unidades; algunas medidas serán enteras, mientras que otras serán decimales.

Todo lo dicho hasta el momento tendría poca utilidad si no se tienen claro cuáles son los indicadores significativos de la calidad de los requisitos. Sobre este tema existen algunos trabajos pioneros [Wilson 97], y abundante literatura en los últimos años [Kasser 04], [Alexander 02], [Fabrini 01]. A continuación se procede a exponer algunos de los indicadores más utilizados, clasificándolos en cuatro categorías principales:

- **Indicadores morfológicos**. Son aquellos, tales como el tamaño, que miden propiedades del texto desde un punto de vista puramente formal, sin considerar su contenido.

- **Indicadores léxicos.** Son aquellos, tales como el número de términos ambiguos, que miden propiedades relativas al contenido del texto y que requieren algún tipo de información de referencia (en este caso, la lista de términos ambiguos definida por el usuario).

- **Indicadores analíticos.** Son aquellos, tales como el uso de formas verbales, que requieren un análisis del texto de los requisitos mediante herramientas lingüísticas relativamente complejas.

- **Indicadores relacionales**. Son aquellos, tales como el número de solapamientos con otros requisitos, que miden propiedades estructurales del conjunto de requisitos, más que propiedades de los requisitos individuales.

2.1.2 Indicadores morfológicos

2.1.2.1 Tamaño.

Las dos medidas más fáciles de obtener sobre un requisito son el tamaño y el índice de legibilidad. El tamaño del requisito puede medirse en caracteres, en palabras, en frases, o en párrafos. Debe notarse que el número de párrafos será poco discriminatorio, ya que será siempre muy pequeño; además, hay que tener cuidado al hacer el recuento, ya que, si el requisito contiene listas enumeradas, el número de párrafos puede resultar engañosamente demasiado grande.

Como ya ha sido comentado, el tamaño adecuado del requisito no será ni muy grande ni muy pequeño, por tanto tendrá una función escalonada convexa. El tamaño del requisito influye directamente en la propiedad deseable de atomicidad, y a través de ella en todas las demás, particularmente en la trazabilidad, verificabilidad y modificabilidad. Nótese que calcular el tamaño requiere que el requisito esté explícitamente identificado. Esto puede parecer obvio hoy día, pero hasta hace pocos años no ha sido siempre así, sino que ha sido frecuente encontrar especificaciones de requisitos que consistían en un texto continuo donde los requisitos no estaban perfectamente determinados y enumerados [Wilson 97].

2.1.2.2 Legibilidad.

Los índices de legibilidad tratan de medir el grado de dificultad de comprensión lectora de un texto. No debe confundirse la legibilidad tipográfica *(legibility*) con la legibilidad lingüística *(readability*), que es la que aquí se hace referencia. Estos índices, a menudo incorporados en los procesadores de texto más habituales, están basados en el promedio de sílabas por palabra (o de letras por palabra, mucho más fácil de medir) y de palabras por frase. La idea es que un texto es tanto más legible cuanto más cortas sean en promedio las frases y las palabras. Por ejemplo, el índice de Flesch responde a la fórmula:

$$LFlesch = 206{,}835 - 1{,}015 \times PPF - (84{,}6 \times SPP)$$

Donde PPF y SPP son respectivamente el promedio de palabras por frase y el promedio de sílabas por palabra. El resultado valora el texto en una escala de 100 puntos; cuanto más alto sea el resultado, más fácil será comprender el documento, por tanto le correspondería una función escalonada creciente. Obviamente, los coeficientes que aparecen en estas fórmulas deben adaptarse a las peculiaridades de cada idioma. En el caso del castellano, José Fernández Huerta hizo ya ésta transposición hace varias décadas, sustituyendo el coeficiente 84,6 por 60,0 en un artículo que, no obstante, no aporta ninguna prueba empírica. El cambio equivale a decir que, para obtener el mismo índice de legibilidad de Flesch citado anteriormente, el número de palabras por frase debe ser igual en castellano y en inglés, pero el número de sílabas por palabra debe ser 1,44 veces mayor en castellano. Otro índice semejante es el de Flesch-Kincaid que responde a la fórmula:

$$LKinkaid = 0{,}39 \times PPF + 11{,}8 \times SPP - 15{,}59$$

Este índice produce un resultado entre 0 y 12, correspondiente al grado escolar indicado para un texto en el sistema de enseñanza norteamericano; por tanto, a diferencia del anterior, le correspondería una función escalonada decreciente: el texto con un índice menor es más legible.

Los índices de legibilidad se han aplicado a los requisitos desde la aparición de los primeros trabajos en métricas de calidad [Wilson 97]. No obstante, el uso de estos índices también ha sido bastante criticado [Kasser 06]. En efecto, tal vez sea esperable que los requisitos estén formados por frases cortas y simples, pero en cambio es completamente normal que, por tratarse de textos técnicos, contengan muchas palabras bastante largas. Así pues, es dudoso que unos índices de legibilidad pensados para calificar lecturas escolares sean aplicables a los requisitos de un sistema informático. En cualquier caso, la propiedad deseable más directamente relacionada con estos indicadores es la comprensibilidad.

2.1.2.3 Puntuación.

Otra medida que es posible considerar, también relacionada con la comprensibilidad de los requisitos, y relativamente fácil de obtener, es el número de signos de puntuación por frase, dividido por la longitud de la frase: la puntuación adecuada es esencial para la comprensibilidad de la frase, y el exceso de puntuación hace el texto más difícil de leer, indicando tal vez que la frase debe ser partida en dos o más frases. La carencia de puntuación en frases largas es igualmente perniciosa, de modo que la función escalonada sería convexa.

2.1.2.4 Acrónimos y abreviaturas.

El exceso de siglas (NASA, E.S.A., etc.) y de abreviaturas (“fra.” por “factura”) también puede utilizarse como indicador de falta de calidad. Las siglas son fácilmente detectables (palabras formadas enteramente, o mayoritariamente, por mayúsculas), aunque tal vez no de forma totalmente determinista El uso de siglas no es pernicioso si se encuentran bien definidas en un glosario de términos; no obstante, el exceso de siglas puede hacer un requisito menos comprensible. Las abreviaturas pueden detectarse como palabras terminadas en “.” que no son final de frase; igualmente, la detección no será determinista (si la abreviatura está al final de la frase, o si la siguiente frase comienza erróneamente por minúscula). El uso de abreviaturas es aún más desaconsejable que el uso de siglas, y normalmente no está justificado.

2.1.3 Indicadores léxicos

Los indicadores morfológicos que han sido descritos en el apartado anterior no requieren de ninguna información adicional proporcionada por el usuario para ser calculados. Por el contrario, el conjunto de indicadores que se agrupan en este apartado y que se denominan “léxicos” se caracterizan por medir propiedades que utilizan información de referencia, es decir, comparan el texto de los requisitos con diversas listas de términos definidas por el usuario. Salvo esta particularidad, se trata de indicadores relativamente sencillos de definir y obtener.

2.1.3.1 Términos conectivos.

En primer lugar, es posible medir la frecuencia de distintos tipos de términos y partículas conectivos que, en general, son necesarios en cualquier construcción lingüística, pero cuyo abuso puede implicar una disminución de la calidad, especialmente en lo que se refiere a las propiedades de atomicidad, precisión, abstracción, comprensibilidad e inambigüedad. En todos ellos la función escalonada de transformación debería ser decreciente, ya que se trata de evitar un abuso, pero el uso moderado es correcto.

- **Número de términos copulativos-disyuntivos.** Estos términos están potencialmente relacionados con la falta de atomicidad de los requisitos. El uso de algunas conjunciones, particularmente “y”, “o”, puede denotar que, más que un requisito, se trata en realidad de dos requisitos distintos (ejemplo: “el usuario se autenticará y visualizará el estado de su cuenta”). Sin embargo, el uso de términos copulativos o disyuntivos puede ser perfectamente legítimo cuando se trata de especificar con precisión una condición lógica, por lo que esta medida debe ser definida y manejada con cuidado (ejemplo: “la caldera se apagará si ha estado encendida más de 10 minutos o si la temperatura del agua es superior a 60ºC”). Por otra parte, los diversos usos de la conjunción “o” (principalmente disyuntivo-inclusivo, disyuntivo-exclusivo, y explicativo) puede originar una falta de precisión, y por tanto ambigüedad o incomprensibilidad.

- **Número de términos negativos.** La acumulación de partículas “no”, “ni”, “tampoco”, “ningún”, “nunca”, “nada”, etc., puede hacer que una frase sea más difícil de comprender, además de incrementar el riesgo de inconsistencias lógicas; afecta especialmente, por tanto, a la propiedad deseable de comprensibilidad.

- **Número de términos de flujo de control**. Son conjunciones o locuciones conjuntivas tales como “mientras”, “cuando”, “si...entonces”, cuyo abuso indica posiblemente un exceso de detalle en la forma de especificar el flujo de control de un proceso o función, traspasando los límites delo que debe ser una especificación abstracta de requisitos. Este indicador está relacionado especialmente con la propiedad deseable de abstracción, y posiblemente con la atomicidad.

- **Número de términos anafóricos.** Son términos que están en lugar de otros términos; típicamente son los pronombres personales (él, ello), relativos (que, donde), demostrativos (éste, ése, aquél), etc. Incluso con un uso gramaticalmente impecable, las anáforas incrementan el riesgo de imprecisiones y ambigüedades en los textos de carácter técnico.

2.1.3.2 Términos imprecisos.

En segundo lugar, algunos de los defectos típicos que conviene eliminar [Alexander 02] están relacionados con el uso de términos imprecisos que por sí mismos introducen ambigüedad en el requisito. Así pues, la comparación con listas de términos “prohibidos” también puede proporcionar métricas bastante interesantes. El uso de términos imprecisos o subjetivos es siempre desaconsejable, por tanto en este caso se hace referencia a funciones escalonadas decrecientes. Es posible agrupar los términos por el tipo de imprecisión que los caracteriza:

- Calidad: “bueno”, “adecuado”, “eficiente etc.

- Cantidad: “bastante”, “suficiente”, “aproximadamente”, etc.

- Frecuencia: “casi siempre”, “generalmente”, “típicamente”, etc.

- Enumeración: “varios”, “en un futuro”, “no limitado a”, etc.

- Probabilidad: “posiblemente”, “probablemente”, “opcionalmente”.

- Usabilidad: “adaptable”, “extensible”, “fácil”, “familiar”, “seguro”, etc.

2.1.3.3 Términos de diseño.

Finalmente, el abuso de términos estrechamente relacionados con el diseño o la tecnología (“método”, “parámetro”, “base de datos”, “applet”) denota falta de abstracción en los requisitos. La función escalonada correspondiente es la decreciente.

2.1.4 Indicadores analíticos

Se denominan “analíticos” a un conjunto de indicadores que requieren un análisis del texto de los requisitos mediante herramientas lingüísticas relativamente complejas y que, en general, no producen resultados completamente deterministas, al igual que en el caso de otros indicadores comentados anteriormente (número de siglas y abreviaturas).

2.1.4.1 Ortografía y gramática.

En primer lugar, es posible obtener el número de errores ortográficos y gramaticales que contenga el texto de los requisitos. Estas dos medidas, conceptualmente simples pero computacionalmente difíciles de realizar, constituyen en todo caso una medida indirecta del cuidado con que están escritos los requisitos.

Además, si el número de errores es elevado, el texto puede llegar a ser impreciso o incluso incomprensible.

2.1.4.2 Tiempo y modo verbales.

En segundo lugar, es posible considerar el uso de los tiempos y modos verbales, que requiere un análisis de la flexión verbal muy dependiente del idioma en que están escritos los requisitos. Desde el comienzo de las investigaciones referenciadas [Wilson 97] se ha subrayado la utilidad de analizar el uso de las formas verbales más típicas para expresar la acción que constituye el núcleo de un requisito: presente o futuro de indicativo (“el usuario visualiza el estado de su cuenta”, “el usuario visualizará el estado de su cuenta”), e infinitivos precedidos de verbos auxiliares de obligación o de posibilidad (en presente o en futuro: “el usuario debe visualizar”, “el usuario tiene que visualizar”, “el usuario podrá visualizar”, “la aplicación permite visualizar”).

El número de formas verbales que significan obligación o posibilidad, denominadas habitualmente en la literatura de modo genérico “formas imperativas” (con imprecisión terminológica que sin embargo respetamos), es un buen indicador de la atomicidad de los requisitos: el exceso de formas imperativas indicaría que el requisito considerado no es suficientemente atómico. Algunos estudios [Wilson 97] pretenden incluso contar los requisitos partiendo del análisis de formas imperativas, cuando ocurre que los requisitos no están ya explícitamente individualizados. Por otra parte, el uso de modo condicional en lugar de futuro de indicativo es un error típico, que probablemente tiene su origen en el deseo inconsciente de expresar un nivel inferior de necesidad en el requisito. Lo correcto es expresar el núcleo del requisito en forma pura, y expresar por separado la necesidad del mismo en forma de atributo (típicamente en tres niveles: esencial, conveniente, opcional). También es necesario considerar aquí el número de usos de la voz pasiva (“es visualizado”) ya pasiva refleja (“se visualiza”): estos usos tienden a dejar el sujeto verbal implícito, lo que conlleva un cierto grado de imprecisión. El análisis de las formas verbales tiene peculiaridades distintas en cada idioma. En el caso del inglés, al que se refiere la mayor parte de la literatura, este análisis puede resultar bastante sencillo, ya que la búsqueda de unas pocas palabras clave proporciona mucha información (shall, must, have/has to, can, should, could, etc.). En el caso del castellano y otras lenguas romances similares, donde la

conjugación verbal es mucho más rica y además existen multitud de excepciones a las reglas gramaticales básicas, el análisis es mucho más costoso.

2.1.4.3 Términos del dominio.

En tercer y último lugar dentro de este apartado, es posible considerar relevante como medida de calidad el número de términos del dominio (tanto sustantivos como verbos) en cada requisito. La obtención de esta medida requiere la normalización de las distintas variantes gramaticales de los términos (género y número en los sustantivos, conjugación de los verbos) para encontrar la forma canónica definida en el dominio; también se requieren herramientas para detectar términos compuestos significativos. Una vez que se han normalizado sustantivos y verbos, se comparan con los términos definidos en el dominio, que puede estar organizado como un simple glosario de términos, o en formas estructuralmente más complejas, tales como tesauros u ontologías. El número de términos del dominio en cada requisito no debería ser ni muy elevado (lo que indicaría pérdida de atomicidad) ni muy escaso (lo que indicaría unos requisitos imprecisos por no usar los términos del dominio, o bien un dominio mal construido, que no recoge suficientemente los términos empleados en los requisitos).

2.1.5 Indicadores relacionales

A diferencia de los indicadores vistos hasta ahora, los indicadores denominados “relacionales” no miden propiedades de los requisitos individuales, sino más bien propiedades estructurales del conjunto de requisitos. Las relaciones entre requisitos, en general, no pueden determinarse automáticamente, por lo que para las medidas de calidad se supondrá que estas relaciones están ya dadas en la representación de los requisitos. Entre otras, se encuentran las medidas relacionales que se exponen en los siguientes sub-apartados

2.1.5.1 Número de versiones de un requisito.

También denominado “volatilidad” del requisito. Un número excesivo de versiones, especialmente si las versiones se suceden en fechas recientes, es una buena indicación de la inestabilidad del requisito, que puede deberse a múltiples causas (mala comprensión de las necesidades del cliente, inseguridad del mismo sobre lo que quiere en realidad, etc.). La calidad de los requisitos exige, entre otras cosas, un alto grado de estabilidad en los mismos, que influye directamente tanto en la validabilidad como en la verificabilidad. El número de versiones es un caso un tanto especial dentro de este apartado, ya que puede considerarse una medida tanto individual como relacional (especialmente si las versiones anteriores también están almacenadas).

2.1.5.2 Grado de anidamiento.

Si los requisitos están estructurados jerárquicamente (ya sea porque unos requisitos están subordinados a otros, o porque están agrupados en paquetes y subpaquetes de requisitos), pueden obtenerse medidas tales como el nivel de profundidad de la jerarquía, y especialmente el grado de anidamiento, es decir, el promedio de elementos que están subordinados a uno dado. Un criterio tradicional de organización de la información, para

facilitar la comprensibilidad, es que el grado de anidamiento no debe ser ni escaso ni excesivo.

2.1.5.3 Número de dependencias de un requisito.

Hacia otros requisitos o, en general, hacia otros artefactos del proceso de desarrollo (se entiende que "B depende de A" cuando B requiere la presencia de A, o cuando un cambio en A puede afectar a B). Del mismo modo que en diseño se busca minimizar el número de dependencias entre artefactos para facilitar el mantenimiento y la reutilización, también debe procurarse minimizar el número de dependencias entre requisitos. El excesivo número de dependencias denota posiblemente falta de atomicidad, comprensibilidad y trazabilidad, afectando por tanto a las tres propiedades finales deseables (validabilidad, verificabilidad y modificabilidad). No obstante, es natural que existan dependencias, y que no estén reflejadas en la representación de los requisitos denotaría no tanto que no existen, sino que el análisis ha sido insuficiente. Por tanto, es posible esperar un número moderado de dependencias por requisito, ni muy grande ni muy pequeño (función escalonada de transformación convexa).

2.1.5.4 Número de solapamientos entre requisitos.

Entendido como número de requisitos que "hablan de lo mismo" que uno dado. Aquí es posible distinguir entre "conflicto" cuando hay contradicción entre dos requisitos, "redundancia" cuando hay una repetición innecesaria (que implica además el riesgo de contradicción), y simple "acoplamiento" cuando no es ninguno de los otros dos casos (y que de alguna manera implica una relación de dependencia). Las herramientas de procesamiento lingüístico (por ejemplo, por coincidencia terminológica, especialmente si hay un dominio de referencia) pueden ayudar a detectar solapamientos entre requisitos, aunque el tipo concreto de solapamiento debe determinarlo el ingeniero. Evidentemente, no deben existir conflictos ni redundancias entre requisitos; en cambio, es natural que haya simples acoplamientos, aunque su número no debería ser excesivo, porque podría denotar falta de atomicidad. Los solapamientos en general afectan a la comprensibilidad, inambigüedad y trazabilidad de los requisitos.

La Tabla 2 resume los indicadores medibles que han sido presentados, junto con las propiedades deseables más directamente relacionadas con cada uno de ellos.

Indicador	**Función**	**Propiedades deseables**										
		Atomicidad	**Precisión**	**Completitud**	**Consistencia**	**Comprensibilidad**	**Inambiguedad**	**Trazabilidad**	**Abstracción**	**Validabilidad**	**Verificabilidad**	**Modificabilidad**
Tamaño	Convexa	X										
Legibilidad	Crec./Decr.					X				*	*	
Puntuación	Convexa					X				*	*	
Siglas y Abrev.	Decreciente					X				*	*	
Conectivos	Decreciente	X	X	*	*	X	X	*	X	*	*	*
Imprecisos	Decreciente		X	*	*	*	X			*	*	
Diseño	Decreciente								X			*
Ort. y Gramática	Decreciente		X	*	*	X	*			*	*	
Imperativos	Convexa	X		*	*	*	*	*	*	*	*	*
Condicionales	Decreciente		X	*	*	*	*			*	*	
Voz pasiva	Decreciente		X	*	*	*	*			*	*	
Dominio	Convexa	X	X	*	*	*	*	*	*	*	*	*
Versiones	Decreciente									X	X	
Anidamiento	Convexa					X				*	*	
Dependencias	Convexa	X		*	*	X	*	X	*	*	*	*
Solapamiento	Decreciente	X		*	*	X	X	X	*	*	*	*

Tabla 2: Indicadores medibles y propiedades deseables relacionadas

2.1.6 Índices de calidad por requisito y medidas globales de calidad

Una vez se han definido los indicadores de calidad que se desea aplicar y sus correspondientes funciones escalonadas de transformación, es posible procesar los requisitos mediante una herramienta automática de gestión de la calidad.

Una herramienta adecuada debe, ante todo, proporcionar indicaciones concretas para mejorar la calidad de los requisitos, distinguiendo entre mandatos para corregir los resultados malos y recomendaciones para mejorar los medios. Por ejemplo, “debe reducir el número de términos imprecisos”, si el resultado ha sido Malo; “conviene aumentar el número de términos del dominio”, si el resultado ha sido Medio; etc. En segundo lugar, la herramienta

debería proporcionar un índice de calidad de cada requisito individual, para lo cual es necesario agregar de alguna manera los resultados obtenidos para cada indicador. Con este índice es posible clasificar los requisitos y así concentrarnos en resolver los problemas de los "peores".

Finalmente, también sería útil disponer de una medida global de la calidad sobre un determinado conjunto de requisitos, lo que requiere agregar los índices de calidad obtenidos para los requisitos individuales. Esta medida global serviría para evaluar la calidad de un proyecto, para apoyar la mejora de la calidad de los requisitos producidos por un ingeniero concreto, etc.

2.1.6.1 Índice de calidad de un requisito

Es razonable pensar que unos indicadores sean más importantes que otros al evaluar el índice de calidad de un requisito, de modo que para calcular una media ponderada será necesario definir el peso relativo de cada indicador, así como asignar valores numéricos a los valores nominales Malo, Medio, Bueno proporcionados por las funciones escalonadas de transformación de cada indicador (por ejemplo, 0-1-2).

Otro enfoque posible consiste en asumir que la medida global no es la media ponderada de los indicadores, sino el valor mínimo de todos ellos: un requisito (o un conjunto de requisitos) es Malo si algún indicador tiene ese valor, aunque los demás tengan un valor superior. Incluso puede adoptarse un enfoque mixto: definir alguno de los indicadores como predominante, lo que significa que, si este indicador es Malo, el resultado global es Malo independientemente de los demás (nótese que esto no equivale a "tomar el mínimo de los indicadores predominantes"); si ningún indicador es predominante y malo a la vez, entonces se toma como medida global la media ponderada.

Por otra parte, no basta con asignar valores numéricos a los valores nominales, sino que es necesario también interpretar el índice de calidad como resultado agregado. Si los valores nominales son traducidos a 0-1-2, como se está asumiendo a modo de ejemplo, entonces el valor agregado (promedio, mínimo o mixto) estará comprendido en el intervalo [0…2]. Dentro de este intervalo se definen a su vez tres sub-intervalos, que no necesariamente serán iguales en tamaño. Por ejemplo, supóngase que el índice de calidad agregado por requisito es Bueno si hay al menos un 50% de indicadores con resultado individual Bueno, al menos un 50% con resultado Medio, y ninguno Malo (u otra distribución equivalente en promedio); y el índice de calidad es Medio si no hay ningún indicador Bueno, al menos un 50% de indicadores con resultado Medio, y el 50% restante con resultado Malo; entonces los sub-intervalos que hay que definir para interpretar el índice de calidad agregado por requisito son: Malo en [0..0,5), Medio en [0,5..1,5) y Bueno en [1,5..2]. Obviamente, se podría establecer un número mayor de intervalos si se considera necesario.

En resumen, una herramienta de medida de calidad basada en indicadores debe definir:

a) si la función de agregación para calcular el índice de calidad por requisito es el promedio ponderado de los indicadores, el mínimo de los indicadores, o el promedio ponderado teniendo en cuenta los indicadores predominantes;

b) los valores numéricos de los valores nominales de los indicadores (Bueno, Medio, Malo), necesarios para calcular el valor agregado;

c) los subintervalos de interpretación de los valores agregados del índice de calidad. Además, por cada indicador utilizado también hay que definir:

- El tipo de función escalonada de transformación (creciente, decreciente, convexa, cóncava).
- Los intervalos de la función escalonada (x_1-x_2 ó x_1-x_2-x_3-x_4, según el tipo de función).
- El peso relativo del indicador para el cálculo del valor agregado.
- Si el indicador es predominante sobre los demás.

El usuario de la herramienta será el encargado de definir todos estos parámetros, según su propio criterio, o el que le imponga la dirección de calidad de la organización. Como puede fácilmente observarse, la mayor dificultad consiste en determinar los intervalos de las funciones escalonadas, así como los pesos relativos de los indicadores, ya que puede caerse fácilmente en la asignación de valores arbitrarios, sin otro fundamento que la "intuición" del usuario.

2.1.6.2 Medida global de calidad de un conjunto de requisitos

Como ya se ha comentado anteriormente, también es útil disponer de una medida global de la calidad sobre un determinado conjunto de requisitos. Esta medida global sirve para evaluar la calidad de un proyecto, de los requisitos producidos por un ingeniero concreto, etc. No obstante, recordemos que el objetivo principal de las mediciones de calidad no es obtener una medida numérica global *(visión del policía)*, sino señalar dónde están los defectos para ayudar a eliminarlos *(visión del consejero)*

Esta medida global es posible calcularla de diferentes formas. Supongamos que los resultados de los indicadores están dispuestos en una matriz de *r* filas (requisitos) y *k* columnas (indicadores). La primera y más evidente forma de calcular la medida global es obtener el índice de calidad por requisito (en horizontal) y luego el promedio para todos los requisitos (en vertical). La segunda forma consiste en obtener el promedio de los valores obtenidos para cada indicador (en vertical), y aplicar entonces la misma función de agregación utilizada para obtener los índices de calidad por requisito (en horizontal). La tercera forma consiste en obtener el promedio de las medidas primitivas asociadas a cada indicador (en vertical), aplicar entonces la función escalonada de transformación, y aplicar entonces la función de agregación (en horizontal). Las tres medidas proporcionan un resultado numérico en el rango [Malo…Bueno], es decir, [0...2] con los valores de ejemplo que se están manejando. Debido a que tanto las funciones escalonadas de transformación

como la función de agregación no son lineales, estos tres resultados no serán iguales en general. El resultado puede interpretarse como Bueno, Medio, Malo, utilizando los mismos intervalos definidos para interpretar el índice de calidad por requisito.

2.2 Organización de Requisitos

Otro de los campos en los que se ahonda en el contenido de este Proyecto Fin de Carrera es la organización de requisitos. Este es un campo mucho más estudiado y del cual existe un mayor número de productos que en el caso de la evaluación y obtención métricas de calidad de requisito. Tal y como se comenta en [Firesmith 03], las soluciones para gestionar los requisitos siguen la siguiente evolución:

- Procesadores de textos.
- Hojas de cálculo.
- Procesadores de texto con hojas de cálculo incrustadas.
- Bases de datos con capacidad de generación de informes.
- Herramientas para la gestión de requisitos
- Herramientas para la gestión de requisitos que soporten más tareas de ingeniería de requisitos que la gestión de requisitos.
- Herramientas de gestión de requisitos que se integren en entornos de desarrollo.

Sin embargo, los productos que han sido desarrollados en este campo no han sido ampliamente aceptados por la industria. Debido a eso, la gran mayoría de analistas siguen utilizando procesadores de textos o herramientas ofimáticas en general para la gestión de los requisitos. Estas herramientas tienen la gran ventaja de que aportan una gran versatilidad y una flexibilidad difícil de batir por el resto de alternativas. Sin embargo, el uso de una aplicación de propósito general hace prácticamente imposible la correcta gestión de los requisitos, desde el punto de visto de la trazabilidad, de identificación de requisitos, revisión y evaluación de los mismos.

La posibilidad de una visualización de la estructura de paquetes, del filtrado de los requisitos y del trabajo colaborativo son características que no pueden ser abordadas por parte de una herramienta tan generalista como un procesador de textos o una herramienta ofimática

Una vez dejado claro la necesidad de una aplicación específica y realizada *ad-hoc* para la gestión y organización de requisitos, es necesario estudiar las alternativas las que se encuentran disponibles en el mercado. Cabe destacar que la mayoría de estos productos o son productos de uso muy limitado debido al coste de su licencia o son productos nacidos dentro del ámbito académico con lo que la fiabilidad y actualización deja mucho que desear. A continuación se realizará una breve descripción de los productos más destacables en este campo:

- **IBM Rational RequisitePro**. Esta aplicación es una de primeras que ha surgido en este campo, y se trata de un producto generado a cargo de la empresa Rational, posteriormente adquirida por *IBM*.

- **IBM Rational DOORS**. Es una de las soluciones líderes del mercado y también pertenece a *IBM* tras la adquisición de su empresa desarrolladora *Telelogic*. Entre sus principales funcionalidades se encuentran las siguientes:

    - Proporciona un entorno fácil para la gestión de requisitos
    - Proporciona acceso vía web a su base de datos de requisitos para una mayor interacción.
    - Permite mantener la trazabilidad desde requisitos a elementos de diseño.
    - Permite mantener la trazabilidad entre requisitos y pruebas
    - Se integra con el resto de productos *CASE (Computer Aided Software Engineering)* de *IBM*.

- **swREUSER**. Se trata de uno de los productos generados por la empresa *TheReuseCompany* [TRC], que entre sus principales funcionalidades, se encuentran las siguientes:

    - Gestión de requisitos: desde la captura automatizada a la difusión de los mismos
    - Estimación mediante Puntos Función y COCOMO II
    - Incluye capacidad de modelado, manteniendo una compatibilidad bidireccional con Rational Rose
    - Control: incluye métricas MOOD y MOOSE
    - Pruebas: gestión de Test Suites y Test Cases
    - Matriz de trazabilidad a todos los niveles
    - Gestión de riesgos:

- **RequirementsStudio.** Se trata una gran aplicación desarrollada en el seno del grupo de investigación *KnowledgeReuse Group*, de la Universidad Carlos III de Madrid, en colaboración con la empresa *TheReuseCompany*. Las principales características de esta herramienta son:

    - Herramienta multiproyecto y multiusuario.
    - Gestión de usuarios y permisos de acceso por proyecto.
    - Registro automático de sesiones de usuario.
    - Atributos de cada requisito: automáticos, obligatorios y opcionales.
    - Atributos y valores desplegables habilitados en función del proyecto y tipo de requisito.
    - Control de versiones de cada requisito.
    - Agrupación de requisitos en paquetes y subpaquetes.
    - Relaciones entre requisitos: traza, subordinación jerárquica, acoplamiento, conflicto, redundancia.
    - Otros artefactos asociados a un requisito: escenarios, modelos...
    - Gestión de reuniones de trabajo asociadas a un requisito.
    - Glosario de términos.
    - Visualización, navegación y edición de requisitos de modo amigable.

- Generación de informes: listado de requisitos en varios formatos, estadísticas, matrices de trazabilidad y consistencia.
- Exportación e importación de proyectos.

Una vez se han comentado las principales opciones de las que dispone un usuario final o una organización a la hora de decidir qué herramienta debe ser utilizada para la organización de requisitos, destaca que, ninguno de ellos, permite una aproximación hacia la evaluación de requisitos, ya que, tal y como ha sido comentado con anterioridad, se trata de una característica poco estudiada y adaptada para el caso que nos atañe. El único ejemplo conocido por nosotros de este tipo de aplicación es *IrQA*, una herramienta generada por la empresa *VisureSolutions* que utiliza a librería CAKEIndexer producida por CISET y que además de la evaluación de requisitos permite las siguientes funcionalidades:

- Organización de requisitos de manera jerarquizada
- Captura de requisitos desde ficheros MS Word, MS Excel y MS Outlook
- Soporte para la definición de dominios
- Soporte para la trazabilidad de requisitos
- Gestión de versiones
- Generación de informes de los requisitos
- Identificación de actores.

A pesar de las grandes funcionalidades que proporciona dicha aplicación existe un conjunto de limitaciones entre las que merece la pena destacar que sólo se permite la definición de un único dominio, así como que las listas de palabras que permiten obtener los valores de ciertos recuentos son fijos, no accesibles por el usuario y no modificables. Del mismo modo, su tasa de acierto en la identificación de verbos y sustantivos no es especialmente elevado, así como la eficiencia en la obtención de los valores asociados a un requisito.

2.3 Algoritmos genéticos

Otra de las partes teóricas en las que ahonda este proyecto es la utilización de algoritmos genéticos para el proceso de obtención de métricas de calidad que permitan evaluar los diferentes requisitos. Este tipo de algoritmos surgieron en los años 1970, de la mano de John Henry Holland [Holland 75] como una de las líneas más prometedoras de la inteligencia artificial.

Son llamados así porque se inspiran en la evolución biológica y su base genético-molecular. Estos algoritmos permiten evolucionar una población de individuos sometiéndola a acciones aleatorias similar a lo que ocurre en la evolución biológica. El objetivo es que esos individuos vayan evolucionando progresivamente en el fin de encontrar una solución para un problema descrito. Debido a eso, es necesario implementar algún criterio similar a la "selección natural" que permitirá decidir cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados.

Un algoritmo genético es un método de búsqueda dirigida basada en probabilidad. Los algoritmos genéticos establecen una analogía entre el conjunto de soluciones de un problema,

llamado *fenotipo*, y el conjunto de individuos de una población natural, codificando la información de cada solución en una cadena, generalmente binaria, llamada cromosoma, tal y como se muestra en la Ilustración 4.

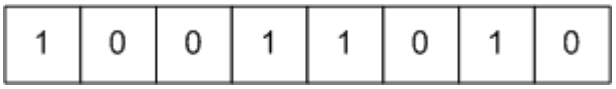

Ilustración 4: Representación binaria de un individuo

Los símbolos que forman la cadena son llamados genes y cuando la representación de los cromosomas se hace con cadenas de dígitos binarios se le conoce como *genotipo*.

Los cromosomas evolucionan a través de iteraciones, llamadas generaciones. En cada generación, los cromosomas son evaluados usando una medida de aptitud o *fitness*, donde son clasificados en función de dicha medida. Las siguientes generaciones se forman utilizando dos operadores genéticos, sobrecruzamiento y mutación, sobre un porcentaje de la población que "mejor se adapta al problema".

Un algoritmo genético puede presentar diversas variaciones, dependiendo de cómo se aplican los operadores genéticos, de cómo se realiza la selección y de cómo se decide el reemplazo de los individuos para formar la nueva población. En general, el pseudocódigo consta de los siguientes pasos:

- **Inicialización**: Se genera aleatoriamente la población inicial, que está constituida por un conjunto de cromosomas los cuales representan las posibles soluciones del problema. En caso de no hacerlo aleatoriamente, es importante garantizar que dentro de la población inicial, se tenga la diversidad estructural de estas soluciones para tener una representación de la mayor parte de la población posible o al menos evitar la convergencia prematura.

- **Evaluación**: A cada uno de los cromosomas de esta población se aplicará la función de aptitud para saber qué tan "buena" es la solución que se está codificando.

- **Condición de parada**. El algoritmo genético se deberá detener cuando se alcance la solución óptima, pero ésta generalmente se desconoce, por lo que se deben utilizar otros criterios de detención. Normalmente se usan dos criterios: correr el algoritmo genético un número máximo de iteraciones (generaciones) o detenerlo cuando no haya cambios en la población. Mientras no se cumpla la condición de término se hace lo siguiente:

    - **Selección.** Una vez se ha evaluado el *fitness* de cada uno de los individuos se procede a elegir aquellos individuos que serán cruzados en la siguiente generación. Los individuos con mejor aptitud tienen mayor probabilidad de ser seleccionados.

    - **Sobrecruzamiento.** El cruzamiento es el principal operador genético, representa la reproducción sexual y opera sobre dos cromosomas a la vez para generar dos descendientes donde se combinan las

características de ambos cromosomas padres. Para cada uno de los genes se selecciona aleatoriamente el de un progenitor.

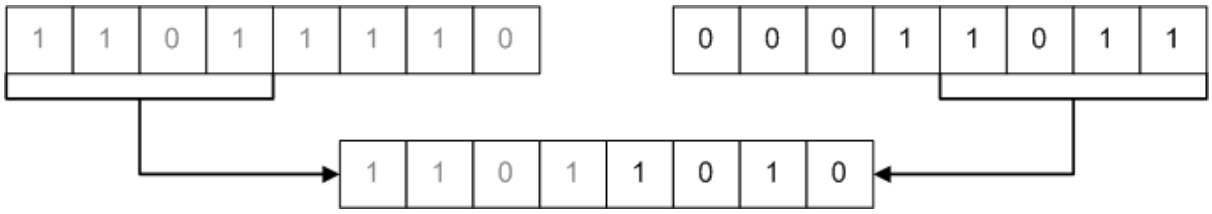

Ilustración 5: Ejemplo de sobrecruzamiento de dos individuos

- **Mutación**. Se modifica aleatoriamente una parte del cromosoma de los individuos. De este modo se permite alcanzar zonas del espacio de búsqueda que no estaban cubiertas por los individuos de la población actual.

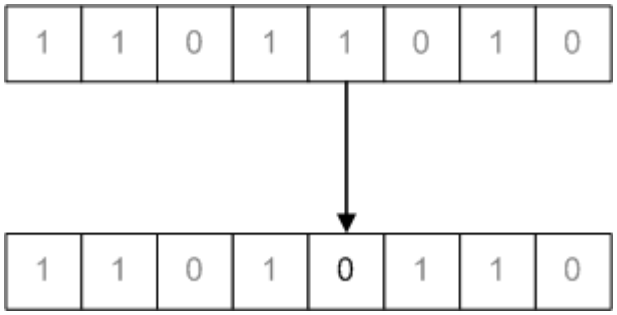

Ilustración 6: Ejemplo de mutación de un individuo

- **Reemplazo.** Una vez, se han aplicado los operadores genéticos, se seleccionan los mejores individuos para conformar la población de la generación siguiente. Para ello se descartan aquellos que tienen un peor nivel de adaptación y se sustituyen por aquellos que han sido generados en el proceso de sobrecruzamiento.

Los algoritmos genéticos tienen una gran cantidad de usos en diferentes disciplinas dada su versatilidad y capacidad de adaptación al entorno, entre dichos campos cabe destacar los siguientes [Tolmos 03]:

- **Optimización.** Se trata de un campo especialmente abonado para el uso de los algoritmos genéticos, por las características intrínsecas de estos problemas. No en vano fueron la fuente de inspiración para los creadores estos algoritmos. Los algoritmos genéticos se han utilizado en numerosas tareas de optimización, incluyendo la optimización numérica, y los problemas de optimización combinatoria.

- **Programación automática.** Los algoritmos genéticos se han empleado para desarrollar programas para tareas específicas, y para diseñar otras estructuras computacionales tales como el autómata celular, y las redes de clasificación.

- **Aprendizaje máquina.** Los algoritmos genéticos se han utilizado también en muchas de estas aplicaciones, tales como la predicción del tiempo o la

estructura de una proteína. Han servido asimismo para desarrollar determinados aspectos de sistemas particulares de aprendizaje, como pueda ser el de los pesos en una red neuronal, las reglas para sistemas de clasificación de aprendizaje o sistemas de producción simbólica, y los sensores para robots.

- **Economía.** En este caso, se ha hecho uso de estos algoritmos para modelar procesos de innovación, el desarrollo estrategias de puja, y la aparición de mercados económicos.

- **Sistemas inmunes.** A la hora de modelar varios aspectos de los sistemas inmunes naturales, incluyendo la mutación somática durante la vida de un individuo y el descubrimiento de familias de genes múltiples en tiempo evolutivo, ha resultado útil el empleo de esta técnica.

- **Ecología.** En la modelización de fenómenos ecológicos tales como las carreras de armamento biológico, la coevolución de parásito-huésped, la simbiosis, y el flujo de recursos.

- **Genética de poblaciones.** En el estudio de preguntas del tipo "¿Bajo qué condiciones será viable evolutivamente un gen para la recombinación?"

- **Evolución y aprendizaje.** Los algoritmos genéticos se han utilizado en el estudio de las relaciones entre el aprendizaje individual y la evolución de la especie.

- **Sistemas sociales.** En el estudio de aspectos evolutivos de los sistemas sociales, tales como la evolución del comportamiento social en colonias de insectos, y la evolución de la cooperación y la comunicación en sistemas multi-agentes.

3 Descripción general de la herramienta

3.1 Perspectiva del producto

Este proyecto es un producto de nueva realización y que, por tanto, no sustituye a ningún producto que se encuentre ahora mismo en implantación. Sin embargo, tal y como se ha comentado, este proyecto parte de la fusión de las características básicas *RequirementsStudio 2.1* [Alonso 08] y *MeCaReq 1.0* [Fernandez 08]. Esto permite ofrecer no sólo una mayor cantidad de funcionalidades sino mejorar la experiencia de usuario especialmente en el ámbito de la evaluación de requisitos y la obtención de métricas de calidad.

3.2 Capacidades generales

Este software tiene un conjunto elevado de funcionalidades que le permite convertirse en un gran gestor de requisitos. Entre sus principales características se engloban las siguientes:

- **Gestión de requisitos.** Una de sus principales funcionalidades es la gestión y organización de los requisitos de diferentes proyectos, entre sus capacidades se encuentran las siguientes:
    - Creación, renombrado y eliminación de paquetes.
    - Creación y modificación de requisitos
    - Exportación de informes de requisitos y relaciones en diferentes formatos
- **Evaluación de requisitos.** Otra de sus principales funcionalidades es la evaluación de los requisitos que pertenecen a los proyectos que son gestionados por la aplicación, con el fin de obtener además de una valoración sobre el requisito, se obtienen recomendaciones con el fin de mejorar la calidad del mismo.
- **Generación de métricas de evaluación.** Se permite la obtención tanto de forma manual como automática mediante la aplicación de algoritmos genéticos.

- **Trabajo colaborativo.** Todas las características comentadas anteriores son aplicadas mediante un trabajo colaborativo, en el que diferentes usuarios acceden simultáneamente desde diferentes localizaciones físicas, ya que los datos residen en un servidor que podrá ser accedido desde cualquier otra localización existente.

3.3 Restricciones generales

El sistema, además de las capacidades generales comentadas anteriormente, debe satisfacer un conjunto de restricciones que deben ser tenidas en cuenta por parte del sistema, entre las que se destacan las siguientes:

- El sistema debe ser ejecutando en un entorno Windows XP y versiones posteriores así como desarrollado bajo la plataforma .NET de Microsoft.
- El sistema debe soportar, al menos, un servidor de base de datos Microsoft SQL Server en versión 2005 y posteriores.

3.4 Características de los usuarios

Los usuarios potenciales son los miembros de organizaciones de tecnologías de la información que se encarguen del desarrollo de cualquier tipo de software y que hagan especial hincapié en la Ingeniería de Requisitos. Cada uno de los usuarios posee el mismo nivel cuando accede inicialmente al sistema, los roles y permisos son dependientes de los diferentes objetos gestionados por el sistema.

Dada la naturaleza colaborativa de la aplicación y la existencia en el sistema de diferentes usuarios es necesario establecer un conjunto de roles que limiten y controlen el acceso de los mismos a objetos del sistema. El principal objeto que debe ser analizado son los proyectos del sistema. Sin embargo, existe otro conjunto de objetos que pueden ser accedidos por diferentes usuarios entre ellos: *Dominio, Lista de palabras, Métricas* y *Experimentos*. Todos los permisos seguirán el estilo jerárquico tal y como se muestra en la Ilustración 7. De forma que las funcionalidades habilitadas para un rol también se encuentran habilitadas para un rol superior.

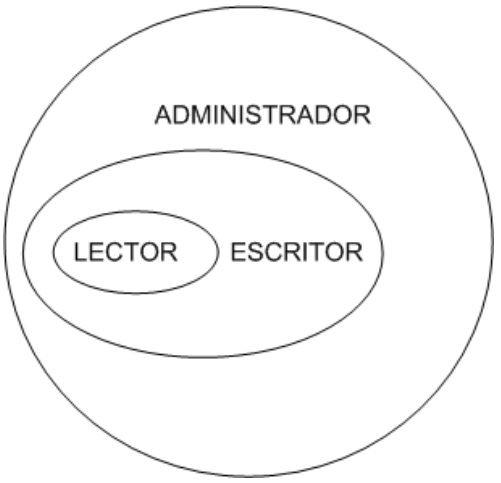

Ilustración 7: Ejemplo de permisos jerárquicos

Centrándonos en un proyecto, existen diferentes roles que delimitan el acceso al proyecto y las posibles acciones que un usuario puede acometer sobre el mismo, tal y como se puede ver en la especificación de requisitos de software que se muestran posteriormente en este documento. Resumiendo, existirán tres tipos de roles respecto a un proyecto:

- **Lector**. Es el usuario con menos permisos en el sistema. Sus posibilidades son básicamente de sólo lectura, no pudiendo añadir ni eliminar ningún tipo de requisito, ni paquetes. Se puede asimilar con un usuario "visitante"

- **Escritor.** Es un usuario con más permisos del sistema que puede no sólo aquellas tareas que se encuentran asignadas al lector, sino también escribir y modificar requisitos así como eliminarlos, crear nuevos paquetes, y similar. Se puede asimilar con un analista del proyecto.

- **Administrador.** Es un usuario con permisos totales dentro del proyecto, pudiendo no sólo trabajar con los requisitos sino con los datos asociados al mismo. Se puede asimilar a un jefe de proyecto.

Desglosando por actividades y haciendo un resumen de lo que se enuncia en los requisitos en *4.2 Requisitos Software*, en la siguiente tabla se muestran una clasificación de las funcionalidades disponibles por cada rol.

	Lector	Escritor	Administrador
Acceder al proyecto	X	X	X
Leer un requisito	X	X	X
Modificar un requisito		X	X
Crear un requisito		X	X
Evaluar requisitos		X	X
Eliminar un requisito		X	X
Generar informes			X
Modificar los datos del proyecto			X
Cambiar los permisos de un usuario			X
Eliminar a un usuario del sistema			X
Eliminar un tipo de requisito			X
Asociar un dominio al proyecto			X
Asociar una métrica al proyecto			X
Desasociar una métrica de un proyecto			X
Desasociar un dominio de un proyecto			X
Crear un tipo relación entre requisitos			X
Eliminar un tipo de relación entre requisitos			X
Exportar un proyecto			X
Importar requisitos en un proyecto			X
Crear paquete		X	X
Renombrar un paquete		X	X
Eliminar un paquete		X	X
Ordenar paquetes		X	X
Mover requisitos		X	X
Duplicar un requisito		X	X
Ordenar requisitos		X	X
Reasignar identificadores a requisitos			X
Cerrar un proyecto			X
Reabrir un proyecto			X

Tabla 3: Correspondencia entre funcionalidades y roles

El resto de objetos del sistema que tienen un acceso compartido, a excepción de los experimentos, siguen el mismo sistema de roles, existiendo dos roles definidos nuevamente de modo jerárquico: *Definidor* y *Usador*.

Un *Usador* es un usuario que sólo tiene acceso de modo lectura al objeto, es decir, se le permite su uso y su incorporación a otros elementos, sin embargo, no pueden realizar modificaciones sobre dichos objetos. Eso queda restringido al rol *Definidor*. Nuevamente, se muestra un desglose en base a las funcionalidades que pueden ser realizadas.

Funcionalidad	Métrica		Dominio		Lista de palabras	
	Definidor	Usador	Definidor	Usador	Definidor	Usador
Añadir un indicador a una métrica	X					
Modificar un indicador de una métrica	X					
Eliminar un indicador asociado a una métrica	X					
Añadir un usuario a una métrica	X					
Eliminar un usuario de una métrica	X					
Modificar los permisos de un usuario sobre una métrica	X					
Modificar la función de agregación de una métrica	X					
Importar/Exportar una métrica	X					
Eliminar una métrica	X					
Duplicar una métrica	X					
Asociar una métrica a un proyecto	X	X				
Añadir un término a un dominio			X			
Eliminar un término de un dominio			X			
Añadir un usuario a un dominio			X			
Eliminar un usuario de un dominio			X			
Modificar los permisos de un usuario sobre un dominio			X			
Importar/Exportar un dominio			X			
Eliminar un dominio			X			
Asociar un dominio a un proyecto			X	X		
Añadir un término a una lista de términos					X	
Eliminar un término de una lista de términos					X	
Añadir un usuario a una lista de términos					X	
Eliminar un usuario de una lista de términos					X	
Modificar los permisos de un usuario sobre un lista de términos					X	
Importar/Exportar una lista de términos					X	
Eliminar una lista de términos					X	
Asociar una lista de términos a un indicador					X	X

Tabla 4: Correspondencia entre funcionalidades de dominio, métricas y lista de términos especiales y roles

Por último, los experimentos siguen un sistema de permisos mucho más simple que todos los casos anteriormente mostrados, en los cuales un usuario sólo tiene un

único tipo de acceso al experimento, de forma que pueda realizar todas las operaciones asociadas al mismo.

3.5 Entorno operacional

Esta aplicación tiene un entorno operacional relativamente sencillo, ya que todo el software necesario para la ejecución del sistema reside en la máquina del cliente. La única excepción es el servidor de base de datos en donde residirán todos los datos relevantes con el sistema. Cabe destacar que la comunicación con la base de datos se realizará mediante el protocolo de comunicaciones TCP (*Transporter Control Protocol*), de forma que su acceso no se encuentra limitado a redes de área local. Sin embargo, es posible utilizar una base de datos local sin necesidad de disponer de conexión a internet.

De ese modo diferentes usuarios pueden acceder desde diferentes dispositivos, pudiendo incluso acceder el mismo usuario desde diferentes ordenadores, a los mismos datos sin necesidad de que deban estar replicados, tal y como se muestra en la Ilustración 8.

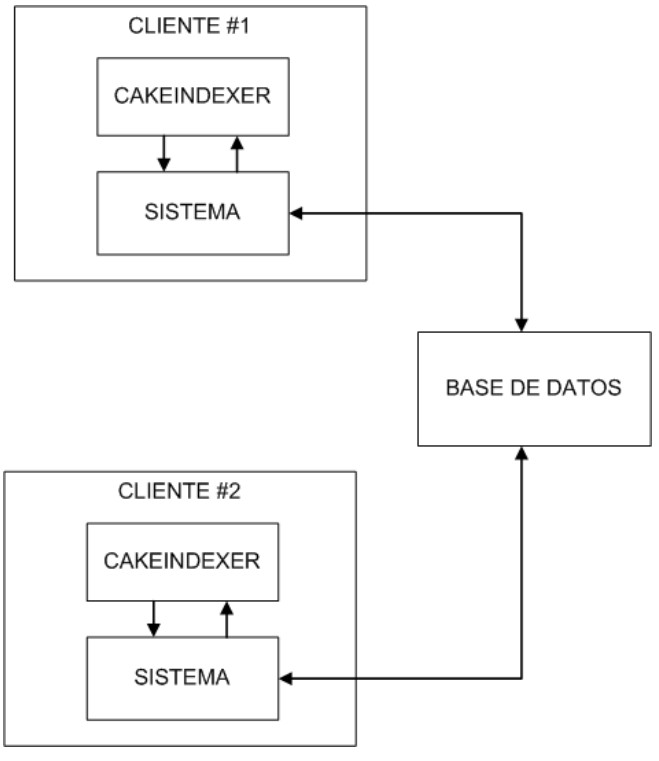

Ilustración 8: Diagrama del entorno operacional

Del mismo modo, otra de las partes relevantes en el entorno del sistema es el módulo externo encargado de obtener los recuentos asociados a la descripción de los requisitos. El módulo recibirá un requisito para que sea analizado y se devolverán los recuentos asociados al mismo. En la versión inicial del proyecto se incluye como módulo de obtención *CakeIndexe*r un módulo desarrollado por parte del Centro de Innovación y Soluciones Empresariales y Tecnológicas [CISET].

3.6 Suposiciones y dependencias

En el ámbito de la realización del proyecto es necesario expresar aquellas dependencias y suposiciones que será necesario realizar. La primera dependencia tal y como ha sido comentada es la existente con un módulo encargado de la obtención de recuentos. En nuestro, caso dado que se utiliza el módulo *CakeIndexer*, implica un conjunto de limitaciones que es necesario comentar y que han sido comentadas más en detalle en [Vazquez 10]:

- **Número de listas de términos limitadas.** Esta librería obliga a disponer de un máximo de 9 listas de términos especiales que sean detectados por ésta. Eso limita el número de indicadores que pueden ser aplicados.

- **Detección de variantes de un sustantivo en cuanto género.** La librería *CAKEIndexer* permite la detección de sustantivos del dominio incluso cuando éstos tienen una variación de número. Sin embargo, en la versión de la librería utilizada, no es capaz de detectar la variante de género en dichos términos.

- **Deficiente detección de verbos y sustantivos**. Como trabajo previo a este Proyecto Fin de Carrera se ha realizado una evaluación de las capacidades de detección de sustantivos y verbos por parte del módulo de evaluación *CakeIndexer*, destacando que su nivel de exactitud no es muy elevado (próximo a un 55%).

3.7 Consideraciones éticas

Al ser un sistema que se encarga de la gestión y el control de la calidad de los requisitos, muchos usuarios pueden ver en él una herramienta de medición de su calidad como analistas o redactores, de forma que su disposición frente al uso de la herramienta puede ser inicialmente adversa. Sin embargo, cabe recordar que el objetivo de la herramienta no es evaluar ni numérica ni nominalmente la calidad de los redactores *(visión del policía)* de los requisitos, sino la de proveer una serie de directrices o consejos que permitan al usuario mejorar la redacción de los requisitos según los criterios establecidos por su jefe de proyecto *(visión del consejero)*.

Del mismo modo, la posibilidad de obtener unas valoraciones numéricas por parte de los administradores de un proyecto de los requisitos realizados por parte de uno o más usuarios no debe ser entendido como un valor real de su calidad con el fin de ser comparado frente a otros compañeros o usuarios, sino un dato estadístico y su uso no debe ser más allá del mismo, no debería ser utilizado con otros fines.

4 Requisitos y Modelo Conceptual

4.1 Modelo conceptual

En esta sección se va a exponer y detallar los diferentes objetos que forman una parte básica del sistema y de su concepción. Dada la gran cantidad de objetos con los que debe trabajar la aplicación, se irá desplegando parte a parte, para facilitar la comprensión por parte del lector.

4.1.1 Gestión de requisitos

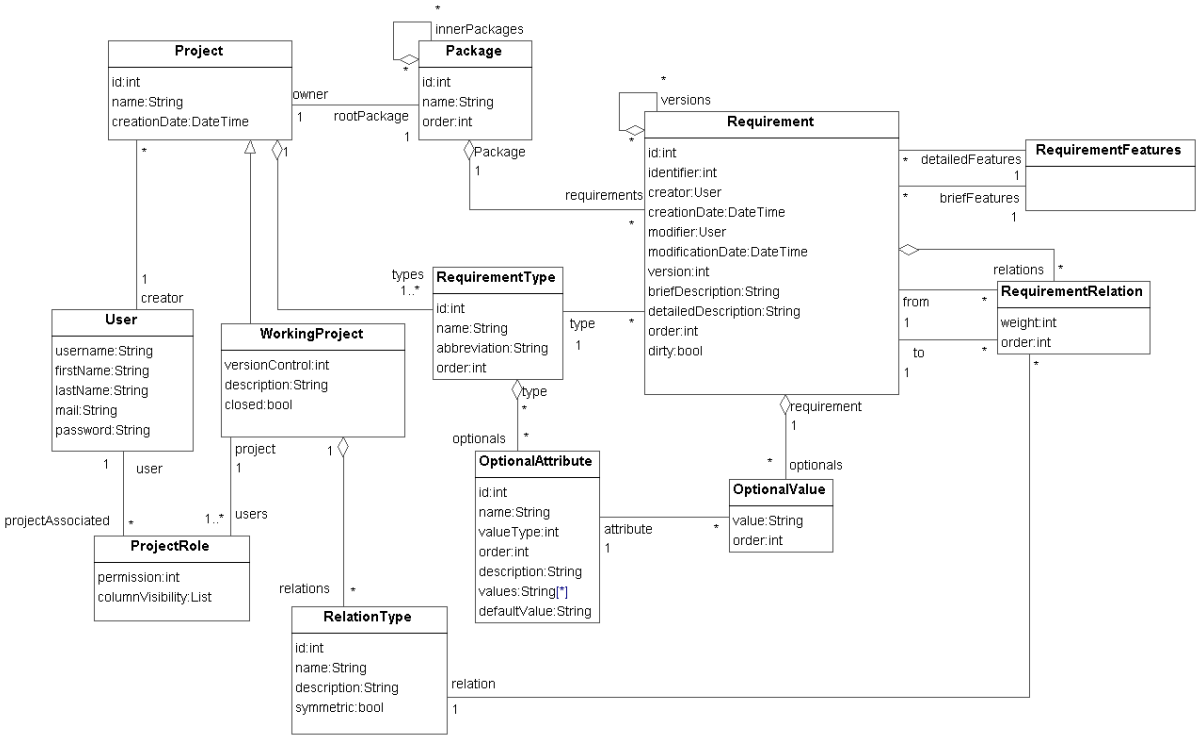

Ilustración 9: Modelo conceptual centrado en Gestión de requisitos

El diagrama de la Ilustración 9 muestra los principales objetos que influyen a la hora de la gestión y organización de requisitos. Los cuales se detallan en la Tabla 5:

Objeto	Explicación
Project	Objeto generalizado de los diferentes tipos de proyecto en la aplicación
WorkingProject	Proyecto básico de la aplicación. Es cada uno de los proyectos que pueden gestionarse por parte del sistema
Package	Objeto que representa cada uno de los paquetes de un proyecto y que son un contender de requisitos
RelationType	Objeto que representa cada una de las relaciones entre requisitos que son válidas o utilizables dentro de un determinado proyecto
Requirement	Objeto que representa cada uno de los requisitos que existen dentro de un paquete.
RequirementType	Objeto que representa un tipo de requisito
RequirementRelation	Objeto que representa una relación entre dos requisitos
OptionalAttribute	Objeto que representa cada uno de los atributos opcionales de los cuales dispone cada uno de los tipo de requisitos
OptionalValue	Objeto que representa cada uno de los valores de los atributos opcionales (*OptionalAttribute*) que tiene cada requisito. Son los datos asociados a la clase intermedia que implementa el concepto clase-asociación (*OptionalAttribute – Requirement*)
ProjectRole	Objeto que representa el rol que cada usuario tiene respecto al proyecto actual. Además mantiene la relación de los atributos que serán visibles en un proyecto por parte del usuario en la visualización del modo tabla de dicho proyecto (atributo *columnVisibility*)
User	Objeto que representa a cada uno de los usuarios del sistema que mantienen un tipo de acceso al proyecto.
RequirementsFeatures	Objeto que encapsula los valores de los recuentos para una determinada descripción de un requisito.

Tabla 5: Clases relativas a la "Gestión de Requisitos"

Un proyecto es un contenedor lógico de requisitos que se encuentran estructurados en paquetes. Existen proyectos que son creados y gestionados por la aplicación (*WorkingProject*) los cuales heredan unas características comunes entre este tipo de proyectos y los proyectos de entrenamiento que se utilizan para la obtención de nuevas métricas de evaluación mediante la aplicación del algoritmo genético, que se encapsulan en la superclase abstracta *Project*.

Cada uno de los proyectos tiene un conjunto de relaciones comunes que deben ser tenidos en cuenta:

- Cada proyecto tiene a su vez un paquete asociado que actúa como raíz de requisitos, que será el punto inicial a partir del cual, empezarán a colgar la diferente estructura de paquetes.

- Cada proyecto tiene asociado al menos un tipo de requisitos. Estos tipos de requisito serán aquellos que podrán tener cada uno de los requisitos de un proyecto (Capacidad, Restricción, Funcional...)

- Cada proyecto tiene asociado un usuario como creador, que es el usuario que introduce, bien creándolo o bien importándolo, el proyecto en el sistema.

- Cada proyecto tiene a su vez un dominio asociado el cual describe aquellos términos que poseen una relevancia especial en el ámbito del proyecto en cuestión.

A su vez sobre todas estas relaciones comunes a todo tipo de proyecto que existe en la aplicación, aquellos proyectos que son gestionados por la misma, es decir, excluyendo aquellos que pertenecen al conjunto de entrenamiento y que son utilizados para tareas de obtención de nuevas métricas, poseen algunas relaciones adicionales que es necesario comentar:

- Cada proyecto gestionado por la aplicación tiene a su vez un conjunto de usuarios que pueden acceder al mismo. Esa asociación se encuentra representada mediante roles, englobados en la clase (*ProjectRole*), cada asociación entre un usuario y un proyecto, se especifica un nivel de acceso (*permission*) que puede tener diferentes valores en función del rol que tenga el usuario con respecto al proyecto.

- Cada proyecto contiene un conjunto de relaciones que permiten definir ciertas vinculaciones existentes entre algunos de los requisitos en el ámbito del proyecto. Existe un conjunto de relaciones por defecto que todo proyecto posee inicialmente (traza, conflicto, redundancia, subordinación jerárquica y acoplamiento) pero pueden ser extendidas bien en el momento de la creación o posteriormente.

- Cada proyecto puede indicar si se trata de un “proyecto activo”, es decir, en el cual se encuentran trabajando o si es un proyecto que ya haya finalizado y se quiere guardar para consulta, como un historial de proyectos. Eso se representa en el atributo *closed* que tomará el valor *true* para proyectos archivados y *false* en aquellos que aún estén activos.

- Cada proyecto tiene un nivel de control de versiones asociados, en el cual se especifica para ese proyecto cómo será el procedimiento cuando se genere una nueva versión o lo que es lo mismo qué ocurrirá cuando se modifique un requisito, las opciones son las siguientes:

    - **Obligatorio.** Cada modificación generará automáticamente una versión nueva y se almacenará la anterior.

    - **Opcional.** Cada usuario decidirá en cada sesión cual desea que sea el comportamiento

    - **Sin control de versiones.** Las modificaciones no generarán nuevas versiones.

A su vez, cada paquete que forma parte del proyecto se convierte en un contenedor de requisitos. Además, cada uno puede contener un conjunto de paquetes internos creando de ese modo una jerarquía de paquetes. Cada paquete se identifica con un nombre y mantiene relación con aquellos requisitos que contiene, así como la jerarquía de paquetes asociada y el proyecto al cual pertenece.

Cada uno de los requisitos especificados en el proyecto, se encuentra ligado a un único paquete y mantiene un identificador único en todo el proyecto. Además posee un conjunto de atributos adicionales que permiten la comprensión de la funcionalidad descrita por el requisito.

Algunos de esos atributos son de “obligada especificación” como pueden ser la descripción corta o descripción larga entre otros. Algunos de estos campos no pueden ser modificados manualmente como el creador, la fecha de creación, el último modificador o la fecha de modificación y su modificación se realizará de manera automática por la aplicación. Cada requisito pertenece obligatoriamente a un tipo y sólo a un tipo de requisito pudiendo “cambiar” el tipo de requisito pero siendo obligatorio que se encuentre ligado a alguno de ellos.

Sin embargo, también existe un conjunto de atributos opcionales. Esos atributos opcionales son dependientes del tipo que tenga asignado el atributo. Existe una relación entre cada uno de estos atributos opcionales y un requisito, y los datos de esa relación, es decir, el valor que tiene para ese atributo ese requisito, se encuentra encapsulado en la clase *OptionalValue*. Los atributos opcionales de un requisito vienen definidos por el tipo al cual pertenecen, es decir, los requisitos pertenecientes a un tipo tendrán que rellenar un conjunto de atributos adicionales. Cabe destacar que la restricción de "tuplas no repetidas" en los distintos sitios donde una clase intermedia implementa una clase-asociación no está garantizada por el modelo conceptual, sino por la programación de las clases.

Para cada requisito se guarda su historial de versiones, con el fin de que la información de su evolución a lo largo del proyecto no desaparezca de forma que cada requisito, mantiene una información con aquellas versiones previas del mismo requisito. También existen relaciones con otros requisitos a través de *RequirementRelation* que define la existencia de una vinculación entre dos requisitos en alguna de las relaciones que hayan sido previamente definidas para el proyecto en cuestión.

Cada requisito podrá establecerse como un requisito sucio o sospechoso. El concepto de requisito sospechoso, o sucio, indica que si se modifica un requisito, se marcarán como sospechosos todos aquellos requisitos que estén relacionados con el requisito modificado, con el fin de identificarlos más fácilmente y someterlos a revisión. La “suciedad” se transmite en la dirección marcada por la asimetría de la relación, o en ambos sentidos si la relación en cuestión es simétrica.

Del mismo modo, cada requisito almacena los resultados de los recuentos de las dos descripciones de un requisito, breve y detallada, a través de sendos objetos *RequirementFeatures*

.

4.1.2 Gestión de sesiones

Otro de los aspectos que es necesario cuidar desde aquellas aplicaciones que tienen cierto ámbito colaborativo o empresarial es el control y gestión de las sesiones que inician los usuarios en el sistema.

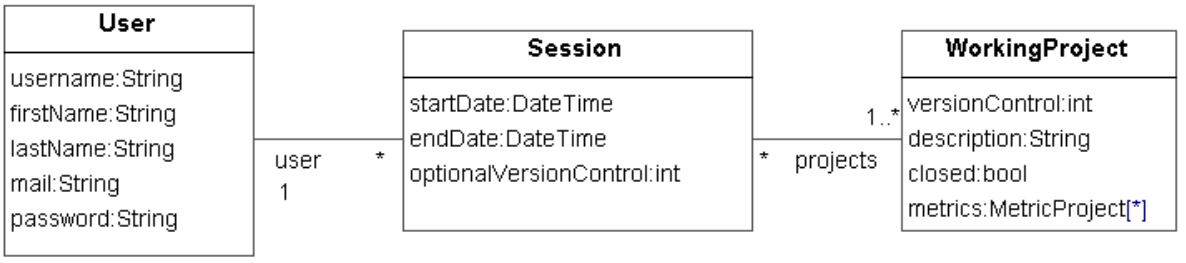

Ilustración 10: Modelo conceptual centrado en Gestión de Sesiones

El diagrama expuesto en la Ilustración 10 muestra el conjunto principal de componentes que influyen a la hora de almacenar y gestionar el historial de acceso de los diferentes usuarios al sistema. Los componentes que influyen en él se muestran en la Tabla 6:

Objeto	Explicación
Session	Objeto que encapsula toda la información referente a la sesión que inicia un usuario en el sistema, tal como fecha de entrada, fecha de salida y versión de control utilizada.

Tabla 6: Clases relativas a la "Gestión de Sesiones"

Dado que el sistema debe almacenar y gestionar todos y cada uno de los accesos que realizan los diferentes usuarios del sistema con el fin de mantener un registro sobre cada una de las sesiones, es necesario establecer una relación entre los usuarios y los proyectos a los cuales puede acceder.

Dado que desde los requisitos se establece que un usuario accede "simultáneamente" a todos sus proyectos no es posible especificar el "momento exacto" de acceso a cada proyecto, sino de acceso general. Para ello, el objeto sesión el cual persiste en la base de datos del sistema almacena cuando un usuario ha iniciado sesión hasta cuando ha estado el usuario en el sistema y qué tipo de control de versiones ha utilizado en caso de que alguno de los proyectos que el usuario tenga asignado el control de versiones opcional. Tal y como se ha comentado anteriormente existe una configuración del proyecto para la cual el usuario decide en cada sesión qué tipo de control de versiones desea. Entre sus opciones estarían:

- **Guardar siempre versión anterior de cada requisito**. Produciría una situación similar al control obligatorio de versiones.
- **Preguntar si guardar versión anterior.** La aplicación consultaría al usuario antes de generar una nueva versión buscando su confirmación.
- **No guardar versión anterior.** Produciría una actuación similar a lo que ocurre cuando el proyecto está configurado con el modo "Sin control de versiones"

Esta situación se produce cuando al menos uno de los proyectos a los cuales tiene acceso el usuario tienen activado el control de versiones opcional.

4.1.3 Gestión de métricas

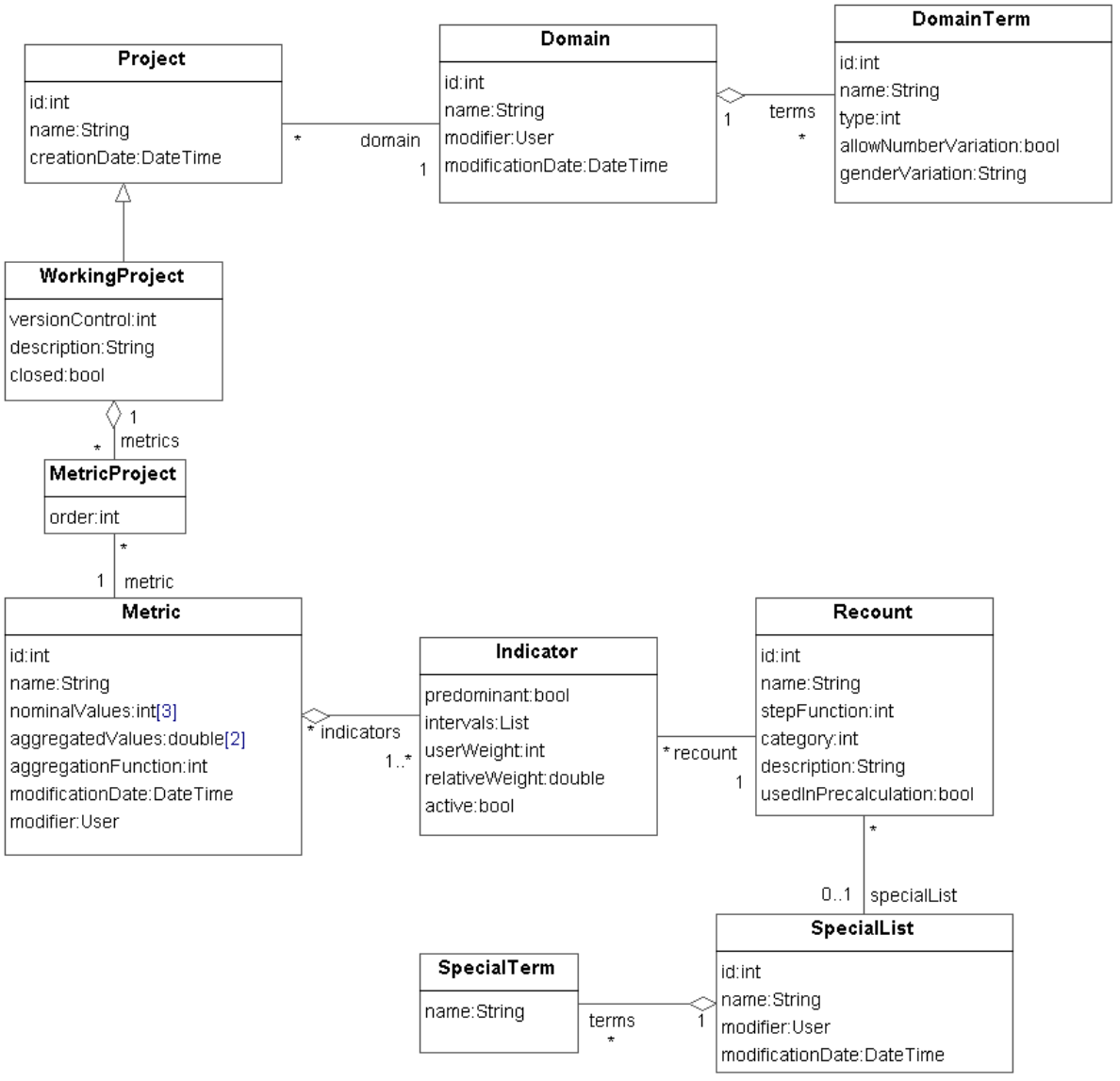

Ilustración 11: Modelo conceptual centrado en "Evaluación de requisitos"

En el diagrama de la Ilustración 11 se muestran los principales conceptos y las relaciones que influyen a la hora de evaluar un determinado proyecto que es gestionado por parte de la propia aplicación. Los principales objetos, que no han sido descritos, se muestran en la Tabla 7:

Objeto	Explicación
Metric	Objeto que representa una métrica que puede ser utilizada para evaluar un determinado proyecto o conjunto de proyectos. Contiene los principales parámetros de configuración así como el listado de indicadores que determinará la calidad de cada uno de los requisitos
Indicator	Objeto que representa cada uno de los atributos variables por los cuales se asocia un recuento a una determinada métrica, es decir, el peso de dicho recuento en la métrica, si es predominante o no, y los intervalos que determinará el valor del requisito.
Recount	Objeto que representa cada uno de los recuentos que pueden ser utilizados por parte de la aplicación
SpecialList	Objeto que representa cada uno de los recuentos que utilizan una lista especial de palabras o de términos.
SpecialTerm	Objeto que representa cada uno de los términos que están asociadas a una lista de tipo *SpecialList.*
Domain	Objeto que representa un determinado dominio asociado a un proyecto
DomainTerm	Objeto que representa cada uno de los términos asociados al dominio
MetricProject	Objeto que representa la relación existente entre un proyecto y una métrica asociada al mismo y la prioridad de la misma.

Tabla 7: Clases relativas a la "Evaluación de requisitos"

El elemento principal para evaluar un proyecto o alguno de sus requisitos es la métrica. Una métrica es un conjunto de criterios de evaluación que se efectúan sobre un conjunto de recuentos. Un recuento no es más que una "característica objetiva" que será evaluada a la hora de determinar la calidad del requisito (Longitud del texto, Número de términos ambiguos…). Dicho recuento, encapsulado en la clase *Recount*, encapsula del mismo modo la información relevante a la categoría del indicador, a través del atributo *category*, así como la función escalonada asociada a través del atributo *stepFunction*. Ambos se representarán mediante un índice numérico[1] que hará referencia a cada uno de los posibles valores explicados en la sección 2.1 Métricas de Calidad en Requisitos.

Del mismo modo existe el atributo *usedInPrecalculation* el cual indica que recuentos se encuentran activos actualmente en el sistema. Esto es necesario porque los proyectos de entrenamiento a diferencia de los proyectos gestionados, no tienen ninguna métrica directamente asociada, y es necesario determinar qué recuentos que se utilizarán para realizar una obtención de los recuentos de los proyectos de entrenamiento en segundo plano. Este problema no existe en el caso de los proyectos gestionados, ya que en proyectos gestionados se hace con los recuentos definidos en la métrica preferente del proyecto. Además no se puede

[1] La utilización de un índice numérico en lugar de un valor nominal es debido a la idea de no introducir textos que sean variables en función del idioma en el cual se encuentre funcionando la aplicación con el fin de que la localización de la aplicación sea una tarea más sencilla.

[2] Esto es así porque es necesario utilizar la API de Word. Por el contrario, para importar proyectos a partir de ficheros con formato Microsoft Excel y Microsoft Access no es necesario tener instaladas estas aplicaciones, ya que se usan conectores ODBC

[3] $F_g' = F_g$ porque el mejor individuo de una generación siempre pasa a la siguiente, salvo que el

considerar que todos estén activos, teniendo en cuenta que existe una limitación de nueve recuentos que tengan asociadas una lista de palabras, de modo que queda como responsabilidad del usuario la decisión de que recuentos se encuentran activos en cada momento.

Algunos de esos recuentos tienen la característica especial de que su valor viene determinado por cuantas palabras de la descripción de un requisito concuerdan con una determinada lista de palabras, lo cual establece una relación opcional entre el recuento definido y la lista de términos especiales contra la cual se realiza dicha comprobación.

Finalmente, un proyecto gestionado por la aplicación tiene un conjunto de métricas asociadas las cuales tienen un orden de "prioridad" o de preferencia dentro del proyecto. Dicha información es encapsulada por el objeto *MetricProject*. Aquella que disponga de una mayor prioridad es considerada como la métrica "*por defecto*", y será la que se utilizará por norma general para evaluar el proyecto y para la cual se optimizarán los tiempos de evaluación.

Cabe destacar que alguno de los recuentos depende del dominio del proyecto determinado. Un dominio se trata de un conjunto de términos que tienen una influencia especial en el ámbito del proyecto del que se está tratando. Por norma general un requisito será mejor si se utilizan más términos asociados al dominio.

4.1.4 Gestión de experimentos

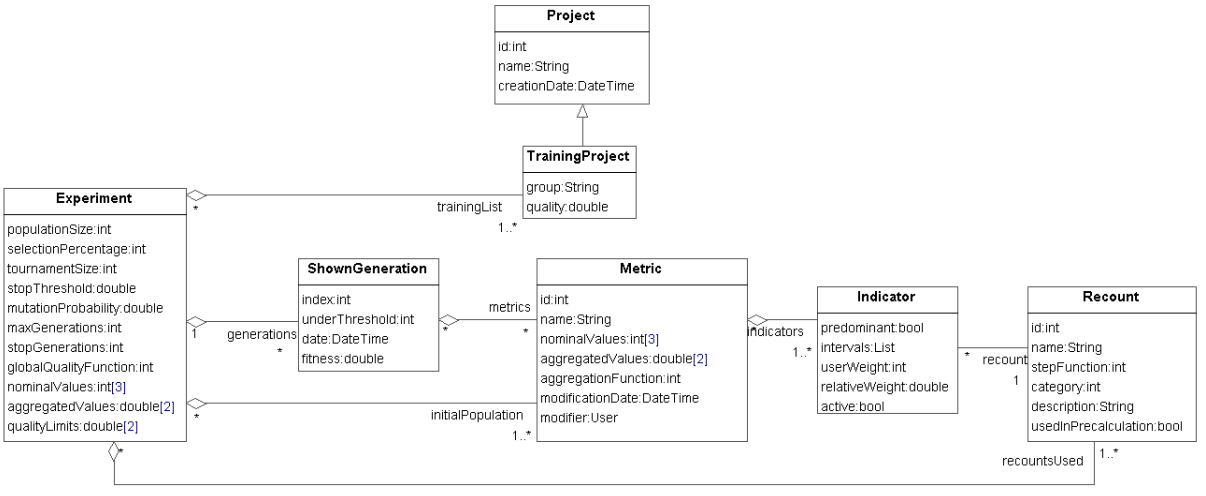

Ilustración 12: Modelo conceptual centrado en "Gestión de Experimentos"

En el diagrama de la Ilustración 12 se muestran los principales objetos que influyen a la hora de generar nuevas métricas. Los principales objetos, que no han sido descritos anteriormente, se muestran en la Tabla 8:

Objeto	Explicación
Experiment	Objeto que encapsula los datos básicos que se han especificado a la hora de comenzar un nuevo experimento para la generación de métricas de evaluación. Entre ellas los parámetros de afinamiento, conjunto de proyectos de entrenamiento utilizados y la población inicial.
TrainingProject	Objeto que encapsula la lógica relacionada con aquellos proyectos que se utilizan como "evaluación humana" y que forman parte del conjunto de entrenamiento
ShownGeneration	Objeto que engloba cada una de las generaciones del experimento en curso que deben ser presentadas al usuario. Contiene la lista de métricas con mejor adaptación en esa generación, y otros datos como la fecha de generación, el *fitness* de la generación, y el número de generaciones consecutivas por debajo del umbral.

Tabla 8: Clases relativas a la "Gestión de Experimentos"

Un usuario puede crear un conjunto de experimentos que son la base para la generación automática de nuevas métricas de entrenamiento. El experimento engloba aquellos parámetros necesarios para que el algoritmo genético pueda converger con el fin de aproximarse lo máximo posible a las calificaciones numéricas de las cuales disponen los proyectos de entrenamiento asignadas manualmente.

4.1.5 Gestión de permisos en los diferentes objetos.

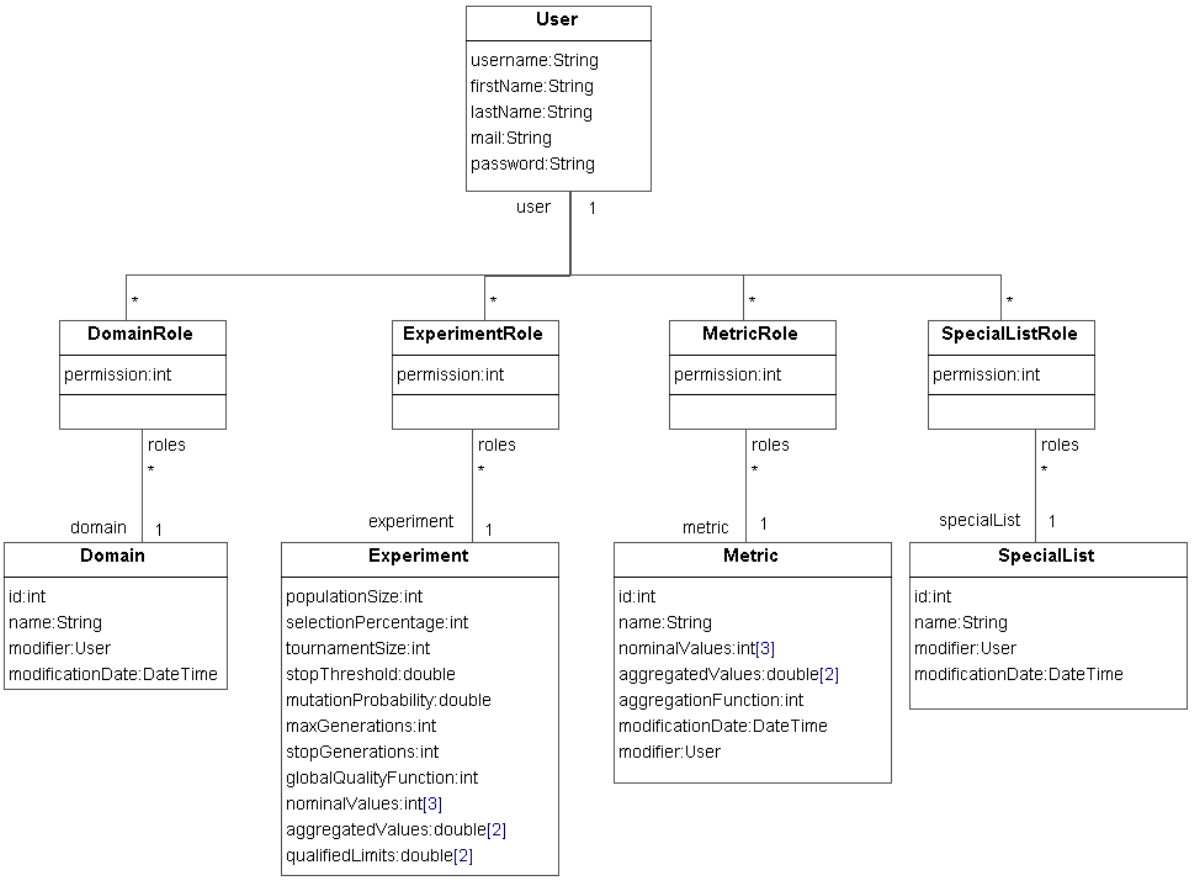

Ilustración 13: Modelo conceptual centrado en "Gestión de Permisos"

En este diagrama se muestran los principales objetos que influyen a la hora de gestionar los permisos sobre los objetos que pueden ser compartidos a excepción de los proyectos. Los principales objetos, que no han sido descritos anteriormente, se muestran en la Tabla 9:

Objeto	Explicación
DomainRole	Objeto que encapsula el acceso de un determinado usuario a un dominio.
SpecialListRole	Objeto que encapsula el acceso de un determinado usuario a una lista especial de palabras
ExperimentRole	Objeto que encapsula el acceso de un determinado usuario a un experimento
MetricRole	Objeto que encapsula el acceso de un determinado usuario a una métrica

Tabla 9: Clases relativas a la "Gestión de Permisos"

Dado que estamos en una herramienta multiusuario y multiproyecto, es necesario que algunos de los objetos puedan compartirse entre diferentes usuarios. Debido a eso, es necesario establecer un acceso mediante roles para los diferentes objetos. Existirá un tipo de rol especial para el dominio, otro para el experimento, otro para la métrica y otro para la lista de palabras. Cada uno de esas relaciones entre cada objeto y el usuario pueden tomar como valor de acceso uno de los dos roles definidos: *Definidor* o *Usador*, tal y como ha sido explicado en *3.4 Características de los usuarios*

4.1.6 Visión general del Modelo Conceptual

Una vez se han explicado y visualizado los principales conceptos que forman el modelo conceptual de la aplicación, ahora se muestra el modelo conceptual de manera completa uniendo las diferentes partes mostradas con anterioridad con el fin de proveer una visión global del mismo:

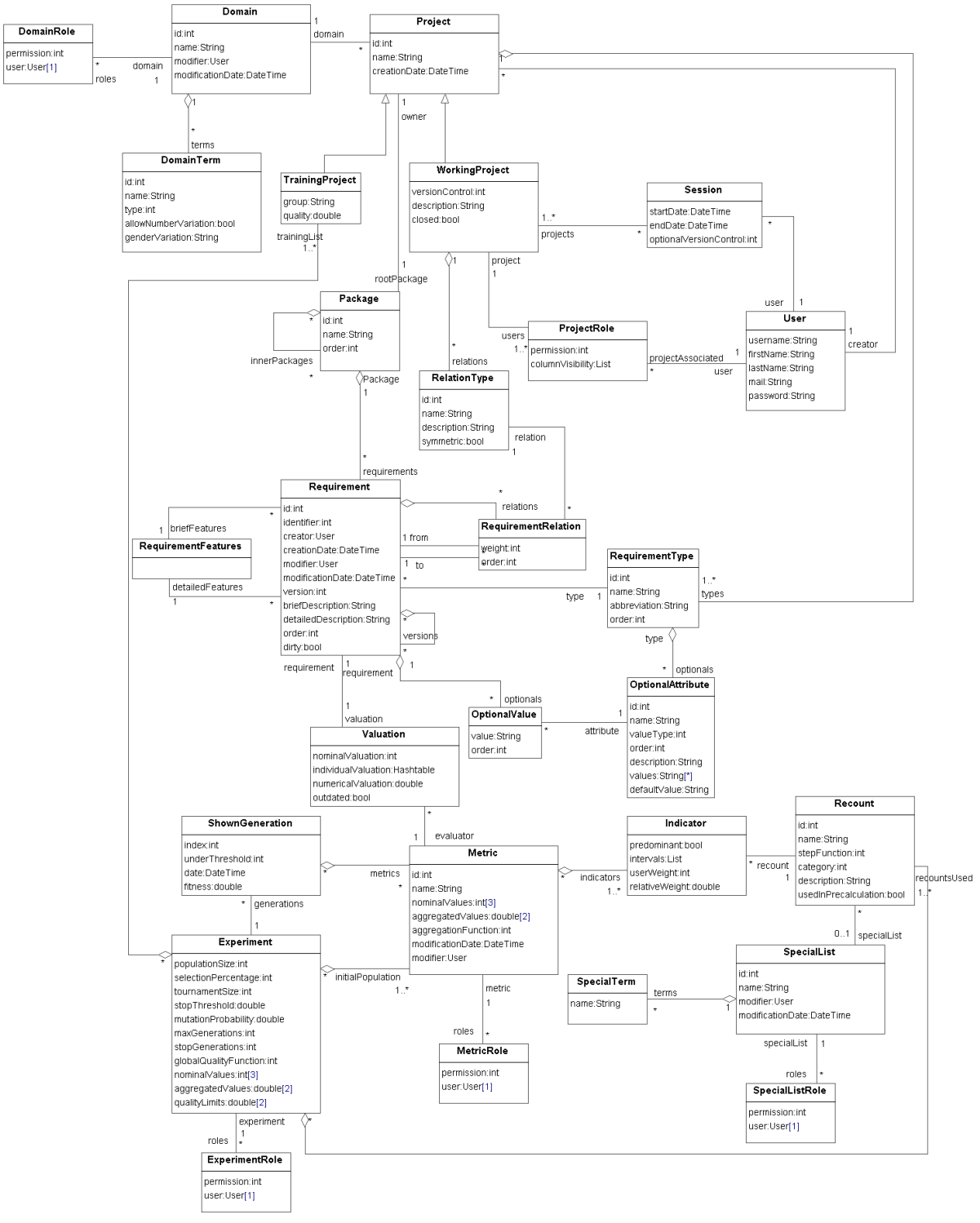

Ilustración 14: Visión general del modelo conceptual

4.2 Requisitos Software

En esta sección se pasarán a especificar los requisitos de software que definen la aplicación, que presentan la siguiente estructura general:

- Requisitos Funcionales
    - Proyectos
        - Paquetes
        - Requisitos
            - Versiones
            - Filtros
            - Relaciones
        - Estadísticas e informes
    - Acceso al sistema
        - Sesiones
    - Métricas
        - Proceso de evaluación
        - Atributos de una métrica
        - Recuentos asociados a una métrica
    - Dominios y Listas de palabras
        - Dominio
        - Lista de palabras
    - Experimentos
        - Proyectos de entrenamiento
        - Proceso de generación de métricas

4.2.1 Requisitos Funcionales

4.2.1.1 Proyectos

Identificador	00001-1	**Tipo**	Funcional
Descripción Breve	Creación de un proyecto		
Descripción Detallada	Un usuario puede crear un proyecto, entendiendo un proyecto como un conjunto de requisitos estructurados en paquetes más una información adicional. Una vez creado el proyecto, el usuario a creador pasará a ser Administrador del mismo.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00002-1	**Tipo**	Funcional
Descripción Breve	Nuevo proyecto. Datos principales		
Descripción Detallada	Entre los datos que un usuario debe proporcionar para proceder a la creación de un proyecto se encuentran los siguientes: - Nombre: Nombre del proyecto, que será una cadena de caracteres entre 1 y 100 caracteres - Descripción: Descripción de un proyecto, será una cadena de caracteres entre 1 y 1000 caracteres, siendo un campo de relleno opcional. - Paquete raíz: Nombre del paquete raíz de requisitos, que será una cadena entre 1 y 100 caracteres. - Control de versiones: Se definirá un nivel del control de versiones asociado al proyecto.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00003-1	**Tipo**	Funcional
Descripción Breve	Nuevo proyecto. Datos automáticos asociados		
Descripción Detallada	Además de los datos que el usuario debe proporcionar para la creación de un proyecto se asignan automáticamente y no son modificables los siguientes: - Creador: Usuario que ha creado el proyecto. - Fecha de Creación: Fecha en la que se ha creado el proyecto.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00004-1	**Tipo**	Funcional
Descripción Breve	Nuevo proyecto. Control de versiones		
Descripción Detallada	El control de versiones de un proyecto se elige a partir de un desplegable. Existen tres posibilidades: • Sin control de versiones: No se guarda ninguna versión anterior del requisito modificado. • Con control opcional de versiones. El usuario decide en cada sesión que opción quiere tomar entre alguna de las disponibles • Con control obligatorio de versiones. Cualquier modificación realizada sobre un requisitos genera una nueva versión		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00005-1	**Tipo**	Funcional
Descripción Breve	Control opcional de versiones		
Descripción Detallada	En caso de que alguno de los proyectos que el usuario tenga abiertos tiene un control de versión opcional, se deberá especificar el tipo de acceso que el usuario desea para dicha versión: - Guardar automáticamente cada versión: Actúa igual que en el modo 'Control de versiones Obligatorio' - No guardar ninguna versión: Actúa igual que en el modo 'Sin control de versiones' - Preguntar tras la realización de cada modificación: Cuando se produzca una modificación, se le preguntará al usuario si desea guardar la versión anterior del mismo		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00006-1	**Tipo**	Funcional
Descripción Breve	Nuevo proyecto. Definición de tipos		
Descripción Detallada	Como segundo paso a la hora de crear un proyecto, el usuario define los tipos de requisitos que desea que estén disponibles cuando se trabaja en ese proyecto. Para ello se debe proporcionar los siguientes datos: • Nombre: nombre que identifique cada uno de los tipos de requisitos que será único en cada proyecto y estará compuesto con una cadena de caracteres de longitud entre 1 y 50 caracteres. • Abreviatura: Cadena de caracteres de hasta un máximo de 3 caracteres que también deberá ser único dentro del proyecto.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00007-1	**Tipo**	Funcional
Descripción Breve	Nuevo proyecto. Habilitación de atributos opcionales		
Descripción Detallada	Como tercer paso para crear un proyecto, el usuario indica qué atributos desea que estén habilitados para cada uno de los tipos de requisitos que definió en el paso anterior. Dispone de una lista de atributos predefinidos, y por cada tipo de requisito marca o desmarca cada uno de estos atributos en función de si desea que esté o no habilitado. Del mismo modo el usuario puede crear nuevos atributos que complementen los ya existentes.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00008-1	**Tipo**	Funcional
Descripción Breve	Nuevo proyecto. Nuevos tipos de atributos opcionales		
Descripción Detallada	El creador de un proyecto, durante el proceso de habilitación de requisitos opcionales puede crear nuevos atributos, para los cuales debe especificar los siguientes datos: - Nombre que deberá ser único en el ámbito del proyecto - Un tipo del atributo, que puede ser 'libre', en caso de que se permita al usuario la entrada de texto libre o 'desplegable' si se permite la elección de un valor entre unos valores predefinidos elegidos a través de una lista desplegable. - Indicar la descripción de dicho atributo, es decir, cuál es su propósito.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00009-1	**Tipo**	Funcional
Descripción Breve	Atributos opcionales definidos por defecto		
Descripción Detallada	El creador de un proyecto dispone, por defecto, de los siguientes atributos opcionales, atributos que puede habilitar o no para cada tipo de requisito definido: • Fuente • Estado • Necesidad • Prioridad • Estabilidad • Claridad • Complejidad • Riesgo • Coste		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00010-1	**Tipo**	Funcional
Descripción Breve	Nuevo proyecto. Valores de un atributo		
Descripción Detallada	El valor de un atributo de un requisito puede introducirse de manera libre en un campo de texto, o puede elegirse, entre una lista de valores, mediante un “desplegable”.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00011-1	**Tipo**	Funcional
Descripción Breve	Nuevo proyecto. Configuración de valores de un atributo		
Descripción Detallada	Una vez seleccionado los atributos opcionales propios de un proyecto, el usuario indica, para cada atributo de los habilitados en el paso anterior y que sean de tipo desplegable los posibles valores que puede tomar.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00012-1	**Tipo**	Funcional
Descripción Breve	Tipos de los atributos opcionales		
Descripción Detallada	De los atributos opcionales disponibles por defecto. Todos son de tipo desplegable a excepción del atributo Fuente, que se trata de un atributo de un atributo de tipo 'libre'		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00013-1	**Tipo**	Funcional
Descripción Breve	Nuevo Proyecto. Asociar usuarios a proyectos		
Descripción Detallada	Como cuarto paso para la creación de un proyecto, se dan de alta a los usuarios que trabajarán en el proyecto. Para ello, se indica que usuarios tendrán acceso a dicho proyecto. Para asociar un usuario a un proyecto, un administrador del proyecto proporciona el identificador de un usuario que tenga una cuenta creada en el sistema y queremos que tenga acceso al mismo. Del mismo modo se debe asignar a dicho usuario un rol en dicho proyecto.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00014-1	**Tipo**	Funcional
Descripción Breve	Roles vinculados a un proyecto		
Descripción Detallada	Existen tres posibles roles que pueden tener los usuarios asociados a un proyecto: - Lector - Escritor - Administrador Los roles son jerárquicos, de modo que todas las acciones que estén disponibles para un rol inferior estarán siempre disponibles para un rol superior.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00015-1	**Tipo**	Funcional
Descripción Breve	Nuevo Proyecto. Dominio y métricas asociado al proyecto		
Descripción Detallada	Como quinto paso para la creación de un proyecto, se especifica, en caso de que sea preciso, el dominio asociado al proyecto. Para ello el creador selecciona el dominio entre aquellos para los cuales tenga acceso. Del mismo modo, se especifica, en caso de que sea preciso, las métricas asociadas al proyecto. Para ello el creador selecciona las métricas entre aquellos para los cuales tenga acceso.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00016-1	**Tipo**	Funcional
Descripción Breve	Nuevo Proyecto. Relaciones asociadas al proyectos		
Descripción Detallada	Como sexto paso para la creación de un proyecto, se especifica, en caso de que sea preciso, las relaciones entre requisitos que podrán ser usadas en el proyecto.		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00017-1	**Tipo**	Funcional
Descripción Breve	Permisos asociados a un 'Lector' de un proyecto		
Descripción Detallada	Si se accede a un proyecto para el cual el rol del usuario sea 'Lector', se pueden consultar pero no editar los contenidos del proyecto. Así, el usuario no puede crear, ni editar requisitos etc.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00018-1	**Tipo**	Funcional
Descripción Breve	Permisos asociados a un 'Escritor' de un proyecto		
Descripción Detallada	Si se accede a un proyecto para el cual el rol del usuario sea 'Escritor', se pueden crear nuevos requisitos o paquetes, editarlos e incluso borrarlos.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00019-1	**Tipo**	Funcional
Descripción Breve	Permisos asociados a un 'Administrador' de un proyecto		
Descripción Detallada	Si se accede a un proyecto para el cual el rol del usuario sea 'Administrador', se puede modificar cualquier dato asociado al proyecto		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00020-1	**Tipo**	Funcional
Descripción Breve	Modificar los permisos de un usuario asociado al proyecto		
Descripción Detallada	Un usuario con rol de 'Administrador' sobre un proyecto puede modificar los permisos del resto de usuarios asociados a un proyecto cuyo rol no sea el de 'Administrador'.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00021-1	**Tipo**	Funcional
Descripción Breve	Almacenamiento de un proyecto en la base de datos		
Descripción Detallada	El proyecto creado, así como todas sus propiedades, se almacena en base de datos. A partir de ese momento, el proyecto estará disponible para que los usuarios que tengan permisos puedan iniciar una sesión y comenzar a trabajar en él.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00022-1	**Tipo**	Funcional
Descripción Breve	Definir un proyecto como abierto o cerrado		
Descripción Detallada	Para un proyecto ya creado, un administrador del proyecto puede indicar si ese proyecto está abierto o cerrado, es decir, si es o no editable. Cuando el proyecto se crea, por defecto estará definido como abierto.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00023-1	**Tipo**	Funcional
Descripción Breve	Creación de Nuevos tipos de relaciones entre requisitos		
Descripción Detallada	El administrador puede definir tipos de relaciones entre requisitos. Para ello deberá proporcionar los siguientes datos: - Nombre de la relación. - Descripción asociada a la relación. - Indicar si la relación es simétrica o no.		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00024-1	**Tipo**	Funcional
Descripción Breve	Creación de nuevos tipos de requisitos		
Descripción Detallada	Un usuario con permisos de 'Administrador' sobre un proyecto puede crear nuevos tipos de requisitos. Para ello se deberán indicar los mismos atributos que en el proceso de creación de un proyecto.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00025-1	**Tipo**	Funcional
Descripción Breve	Modificación de un de datos básicos de un proyecto		
Descripción Detallada	El usuario con permisos de 'Administrador, puede cambiar algunos de los datos especificados durante el proceso de creación del proyecto: - Nombre del proyecto - Descripción del proyecto - Control de versiones del proyecto		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00026-1	**Tipo**	Funcional
Descripción Breve	Modificación de tipos de requisito		
Descripción Detallada	Un usuario que tenga permisos de 'Administrador' sobre un proyecto puede indicar en cualquier momento la modificación de alguno de los datos de un tipo de requisitos asociado al proyecto.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00027-1	**Tipo**	Funcional
Descripción Breve	Modificación de dominio asociado al proyecto		
Descripción Detallada	Un usuario con permisos de 'Administrador' sobre un proyecto puede cambiar el dominio que pasa a estar asociado al proyecto.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00028-1	**Tipo**	Funcional
Descripción Breve	Invalidación de recuentos debido al cambio de dominio asociado al proyecto		
Descripción Detallada	Cuando se realice un cambio de dominio asociado a un proyecto, automáticamente los recuentos que tengan asociados esos requisitos se verán invalidados y será necesario volver a calcularlos.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00029-1	**Tipo**	Funcional
Descripción Breve	Modificación de métricas asociadas al proyectos		
Descripción Detallada	Un usuario con permisos de 'Administrador' sobre un proyecto puede añadir o eliminar las métricas asociadas al mismo.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00030-1	**Tipo**	Funcional
Descripción Breve	Eliminar un tipo de requisito		
Descripción Detallada	Un usuario que tenga permisos de 'Administrador' sobre un proyecto puede indicar en cualquier momento la supresión de un tipo de requisitos asociado al proyecto, siempre que no exista ningún requisito asociado a dicho tipo.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00031-1	**Tipo**	Funcional
Descripción Breve	Eliminar usuarios asociados a un proyecto		
Descripción Detallada	Un usuario con permisos de 'Administrador' sobre un proyecto puede eliminar a un usuario asociado a un proyecto. A partir de ese momento el usuario no podrá acceder al sistema. Sin embargo, se mantendrán sus relaciones con los requisitos que así lo requieran.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00032-1	**Tipo**	Funcional
Descripción Breve	Eliminar un proyecto		
Descripción Detallada	Un usuario que tenga permisos de 'Administrador' sobre un proyecto, puede indicar en cualquier momento el borrado de dicho proyecto. En ese momento se eliminará toda la información asociada a dicho proyecto.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00033-1	**Tipo**	Funcional
Descripción Breve	Importación y exportación de requisitos en un proyecto existente		
Descripción Detallada	Un usuario con rol 'Administrador' sobre un proyecto, puede importar y exportar de requisitos en un proyecto existente mediante un fichero XML que cumple un determinado XML Schema		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00034-1	**Tipo**	Funcional
Descripción Breve	Importación de proyectos		
Descripción Detallada	Un usuario puede importar un nuevo proyecto a través de los diferentes formatos: - Fichero XML propio de la aplicación ReqStudio Plus. - Fichero MDB propio de la aplicación ReqStudio 2.1 - Fichero TXT que cumple con el formato especificado en II.1.3.5 Formato Fichero de texto plano - Fichero DOC etiquetado que cumple con el formato especificado en II.1.3.3 Formato Microsoft Word 2003 y 2007 - Fichero XLS etiquetado que cumple con el formato especificado en II.1.3.2 Formato Microsoft Excel 2003 y 2007 El usuario importador automáticamente pasa a ser creador del proyecto y administrador del mismo.		
Fuente	Tutor	**Necesidad**	Deseable

4.2.1.1.1 Paquetes

Identificador	00035-1	**Tipo**	Funcional
Descripción Breve	Estructura de paquetes		
Descripción Detallada	La aplicación ofrece la estructura “paquete” como un contenedor de requisitos. Es una estructura jerárquica, en forma de árbol, de manera que un paquete puede contener tanto requisitos como otros paquetes. Por defecto existe, en todo proyecto, un paquete raíz con el nombre dado al crear el proyecto.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00036-1	**Tipo**	Funcional
Descripción Breve	Creación de paquetes		
Descripción Detallada	Un usuario con permisos de 'Escritor' o 'Administrador' sobre un proyecto puede crear un paquete. Para crear un paquete basta con darle un nombre. El paquete creado tendrá como paquete padre aquel paquete que estuviera seleccionado en el momento de su creación. El paquete se crea vacío, es decir, no tendrá paquetes hijos ni contendrá requisitos. El nombre no poda repetirse entre el resto de subpaquetes que compartan el mismo paquete padre.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00037-1	**Tipo**	Funcional
Descripción Breve	Renombrado de paquetes		
Descripción Detallada	Un usuario con permisos de 'Escritor' o 'Administrador' sobre un proyecto puede decidir renombran un paquete. Se debe comprobar nuevamente que el nombre no se repita entre los paquetes hijos del paquete padre del paquete renombrado.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00038-1	**Tipo**	Funcional
Descripción Breve	Eliminar un paquete		
Descripción Detallada	Un usuario que contenga un rol de 'Escritor' o 'Administrador' sobre un proyecto puede determinar el borrado de un paquete. En ese momento se eliminará no sólo el paquete, sino que se procederá a eliminación de todos sus paquetes hijos así como los requisitos de una forma no recuperable. Se producirá un borrado en cascada		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00039-1	**Tipo**	Funcional
Descripción Breve	Ordenación de paquetes		
Descripción Detallada	Cualquier usuario con rol 'Escritor' o 'Administrador' puede establecer un orden lógico entre los paquetes dependientes del mismo paquete padre. Dicho orden será el mismo para todos los usuarios que tengan acceso al mismo y se mantendrá de manera permanente.		
Fuente	Tutor	**Necesidad**	Deseable

4.2.1.1.2 Requisitos

Identificador	00040-1	**Tipo**	Funcional
Descripción Breve	Creación de un nuevo requisito		
Descripción Detallada	Un usuario con un rol 'Escritor' o 'Administrador' en el proyecto puede crear un requisito. El requisito creado pertenecerá al paquete seleccionado en el momento de la creación del requisito. Se mostrará una ficha vacía para que el usuario la rellene.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00041-1	**Tipo**	Funcional
Descripción Breve	Eliminación de un requisito		
Descripción Detallada	Un usuario con un rol 'Escritor' o 'Administrador' en el proyecto puede eliminar un requisito. Al eliminar un requisito, se eliminan todas sus relaciones, todas sus versiones.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00042-1	**Tipo**	Funcional
Descripción Breve	Atributos de un requisito		
Descripción Detallada	Un requisito viene definido por una serie de campos o atributos. Algunos de estos atributos son obligatorios y otros opcionales. Dentro de los obligatorios, unos son generados por la aplicación de manera automática y otros los define el usuario.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00043-1	**Tipo**	Funcional
Descripción Breve	Atributos obligatorios, automáticos y no modificables de un requisito		
Descripción Detallada	Hay una serie de atributos que son obligatorios para cualquier requisito que se cree, y además son generados de manera automática por la aplicación. No son modificables. Son los siguientes: • Identificador. • Versión • Creador. • Fecha de creación. • Último modificador. • Fecha de última modificación.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00044-1	**Tipo**	Funcional
Descripción Breve	Atributos obligatorios y modificables de un requisito		
Descripción Detallada	Hay una serie de atributos que son obligatorios para cualquier requisito que se cree, y cuyo valor ha de ser introducido por el usuario. Son los siguientes: - Tipo: se elige entre una lista de valores, lista que habrá sido definida al crear el proyecto. - Descripción breve: campo de texto en el que el usuario puede introducir una breve descripción acerca del requisito. - Descripción detallada: campo de texto en el que el usuario puede introducir una descripción completa del requisito.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00045-1	**Tipo**	Funcional
Descripción Breve	Identificador de un requisito		
Descripción Detallada	Cada requisito creado en un proyecto tiene un identificador único en el proyecto. Este identificador se crea, de manera automática concatenando, mediante un guión, un número entero que se irá autoincrementando y será único en todo el proyecto y la versión del requisito.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00046-1	**Tipo**	Funcional
Descripción Breve	Visualización de requisitos		
Descripción Detallada	Un usuario puede visualizar un requisito de dos maneras diferentes. Una mediante una tabla en la que aparecen todos los requisitos del paquete seleccionado y algunas de sus propiedades, es decir, un listado o catálogo de los requisitos del paquete seleccionado. La segunda posibilidad es, a partir de la vista anterior, “pinchar” en un requisito determinado, con lo que se despliega una ficha en la que pueden visualizarse todas sus propiedades.		

Fuente	Tutor	**Necesidad**	Esencial

Identificador	00047-1	**Tipo**	Funcional
Descripción Breve	Visualización de requisitos. Modo tabla		
Descripción Detallada	En la vista en forma de tabla, se muestran inicialmente todos los atributos habilitados de un requisito. Pero en cualquier momento, el usuario puede elegir qué atributos quiere o no quiere mostrar.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00048-1	**Tipo**	Funcional
Descripción Breve	Ocultación de atributos de requisitos en modo tabla		
Descripción Detallada	El usuario puede mediante una ventana de diálogo para tal fin, que atributos desea que sean ocultados respecto a los requisitos del proyecto actual. Dichas elecciones serán persistentes entre diferentes sesiones y son propios de cada usuario con acceso al proyecto		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00049-1	**Tipo**	Funcional
Descripción Breve	Visualización de requisitos. Modo ficha		
Descripción Detallada	Es el modo de visualización más detallado. Se despliega una ficha con diferentes pestañas. En la pestaña principal se muestran todos los atributos habilitados para el requisito en cuestión. Algunos serán modificables y otros no. El resto de pestañas muestran un listado de requisitos relacionados y un listado de versiones anteriores del requisito.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00050-1	**Tipo**	Funcional
Descripción Breve	Ficha de un requisito. Navegación		
Descripción Detallada	A partir de la ficha de un requisito, se puede navegar al requisito siguiente, al anterior, al primero y al último. Esta navegación se produce sobre el conjunto de requisitos del paquete actual. El orden de la navegación coincide con el orden en que aparecen los requisitos en la vista en forma de tabla.		
Fuente	Tutor	**Necesidad**	Opcional

Identificador	00051-1	**Tipo**	Funcional
Descripción Breve	Ficha de un requisito. Varias visualizaciones		
Descripción Detallada	Se puede ver de manera simultánea la ficha de dos requisitos independientemente de que sean del mismo o de distinto paquete		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00052-1	**Tipo**	Funcional
Descripción Breve	Editar atributos de un requisito		
Descripción Detallada	Un usuario con el rol de 'Escritor' o 'Administrador' sobre el proyecto puede editar los atributos que sean modificables de un requisito desde cualquiera de las dos vistas señaladas, siempre y cuando el proyecto no esté definido como cerrado		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00053-1	**Tipo**	Funcional
Descripción Breve	Mover requisitos dentro de paquetes		
Descripción Detallada	Un usuario con permisos de 'Escritor' o 'Administrador' puede mover requisitos que se encuentren en un paquete a otro especificando el paquete destino. Se puede mover más de un requisito en una sola acción, no siendo necesaria una acción para cada requisito		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00054-1	**Tipo**	Funcional
Descripción Breve	Duplicación de requisitos		
Descripción Detallada	Un usuario puede duplicar un requisito desde la vista en forma de tabla. Como consecuencia de esta acción se crea un nuevo requisito en el mismo paquete que el duplicado. Se mantienen los valores del requisito original a excepción de los siguientes: - El identificador de este requisito es nuevo, para no confundirlo con el anterior. - Además, la fecha de creación será la del momento de la duplicación y el autor será el autor de la duplicación. - La versión de este requisito será “1”.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00055-1	**Tipo**	Funcional
Descripción Breve	Ordenación de requisitos		
Descripción Detallada	En la vista en forma de tabla, la aplicación ofrece la posibilidad de ordenar los requisitos en función de las propiedades que se muestran en esa vista. Podrán ordenarse en orden ascendente o descendente.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00056-1	**Tipo**	Funcional
Descripción Breve	Ordenación lógica de requisitos		
Descripción Detallada	La aplicación ofrece la posibilidad de que un usuario defina su propia ordenación lógica de requisitos dentro de un paquete. Para ello, en la vista en forma de tabla, puede "subir" o "bajar" los requisitos de posición en la tabla, para ordenarlos a su gusto. Este orden se mantendrá de una sesión a otra.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00057-1	**Tipo**	Funcional
Descripción Breve	Reasignación de identificadores de requisitos		
Descripción Detallada	En la vista en forma de tabla, la aplicación ofrece la posibilidad de realizar una nueva reasignación de los identificadores de los requisitos. En el caso de utilizar esta opción, se cambiará los valores de los identificadores de los requisitos y se comenzarán a asignar comenzando en 1, siguiendo el orden de paquetes y de requisitos disponibles en la aplicación		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00058-1	**Tipo**	Funcional
Descripción Breve	Papelera de requisitos		
Descripción Detallada	Existirá una papelera de requisitos donde irán por defecto los requisitos que sean eliminados de manera individual. Es persistente durante sesiones e independiente para cada usuario. Sólo un usuario que ha borrado un requisito puede 'restaurarlo'. El usuario puede en cualquier momento determinar el vaciado de la papelera de requisitos y esos requisitos se borrar definitivamente.		
Fuente	Tutor	**Necesidad**	Opcional

4.2.1.1.2.1 Versiones

Identificador	00059-1	**Tipo**	Funcional
Descripción Breve	Versiones de un requisito		
Descripción Detallada	La aplicación ofrece la posibilidad de guardar versiones antiguas de los requisitos a medida que estos van cambiando, según el modo de control de versiones del proyecto y de la sesión.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00060-1	**Tipo**	Funcional
Descripción Breve	Visualización de versiones de un requisito		
Descripción Detallada	Para un requisito se puede visualizar, desde la última pestaña de su ficha un listado con todas sus versiones antiguas. Por cada versión se indica su fecha de creación y su autor. Desde aquí pueden verse más detalles de cada versión antigua accediendo a su ficha. La ficha se abre en modo de sólo lectura y no permite navegar por las pestañas, sólo nuestra los atributos del requisito.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00061-1	**Tipo**	Funcional
Descripción Breve	Recuperación de versiones de un requisito		
Descripción Detallada	Un usuario con rol de 'Escritor' o 'Administrador' sobre el proyecto puede recuperar una versión anterior de un requisito a través de la pestaña de Versiones, en la ficha del requisito. En ese momento se genera una nueva versión y se copian los datos de la versión que queremos recuperar. El resto de versiones seguirá disponible para su visualización y recuperación.		
Fuente	Tutor	**Necesidad**	Esencial

4.2.1.1.2.2 Filtros

Identificador	00062-1	**Tipo**	Funcional
Descripción Breve	Filtrado de requisitos		
Descripción Detallada	Un usuario puede crear un conjunto de filtros para utilizar sobre los requisitos. Para crear un nuevo filtro, se elige la propiedad por la que se quiere filtrar, el operador de filtrado y el valor con el que queremos filtrar.		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00063-1	**Tipo**	Funcional
Descripción Breve	Persistencia de los filtros		
Descripción Detallada	Los filtros se encuentran almacenados en la base de datos y están asociados a un único usuario.		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00064-1	**Tipo**	Funcional
Descripción Breve	Filtrado de requisitos. Operadores		
Descripción Detallada	Los operadores disponibles para el filtrado de requisitos son los siguientes: - igual a - distinto de		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00065-1	**Tipo**	Funcional
Descripción Breve	Habilitación y aplicación de filtros		
Descripción Detallada	Cuando se crea un filtro, no se habilita por defecto. El usuario puede habilitar los filtros que se considere oportuno mediante la aplicación. En el momento que se indique la aplicación de los filtros, se aplicarán de manera conjunta todos los filtros que se encuentren habilitados en el momento de la ejecución.		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00066-1	**Tipo**	Funcional
Descripción Breve	Búsquedas de texto de requisitos		
Descripción Detallada	Un usuario, desde la vista de requisitos en forma de tabla, puede buscar un fragmento de texto. La aplicación busca este fragmento de texto sobre cada una de las propiedades de los requisitos actualmente mostrados en la tabla. Sólo se busca en aquellas propiedades que están visibles en el momento de la búsqueda.		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00067-1	**Tipo**	Funcional
Descripción Breve	Reemplazo de texto en requisitos		
Descripción Detallada	La aplicación permite, desde la vista en forma de tabla, la posibilidad de realizar un reemplazamiento global de un texto por otro.		
Fuente	Tutor	**Necesidad**	Opcional

4.2.1.1.2.3 Relaciones

Identificador	00068-1	**Tipo**	Funcional
Descripción Breve	Relaciones entre requisitos		
Descripción Detallada	Un usuario con permisos de 'Escritor' o 'Administrador' puede definir relaciones entre los requisitos. Estas relaciones pueden verse y editarse desde la segunda pestaña de la ficha de un requisito. Aquí se mostrará un listado de requisitos relacionados. Por cada requisito relacionado se muestra su identificador, su descripción breve, el tipo de relación y el peso de la relación.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00069-1	**Tipo**	Funcional
Descripción Breve	Borrado de relaciones entre requisitos		
Descripción Detallada	Un usuario con rol 'Escritor' o 'Administrador' sobre el proyecto puede borrar relaciones entre los requisitos. Estas relaciones pueden borrarse desde la segunda pestaña de la ficha de un requisito.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00070-1	**Tipo**	Funcional
Descripción Breve	Relaciones por defecto entre requisitos		
Descripción Detallada	Las relaciones que la aplicación ofrece son: • Trazabilidad: típicamente, relaciones entre requisitos de usuario y requisitos software. Relación anti-simétrica. • Subordinación jerárquica. Relación anti-simétrica. • Acoplamiento. Relación simétrica. • Conflicto. Relación simétrica. • Redundancia. Relación simétrica.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00071-1	**Tipo**	Funcional
Descripción Breve	Peso de la relación entre requisitos		
Descripción Detallada	Cuando se define una relación entre dos requisitos, ha de indicarse el "peso" de esta relación. Podría definirse este concepto como la importancia o fuerza que posee la relación. Los valores serán 'Bajo', 'Medio' o 'Alto'.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00072-1	**Tipo**	Funcional
Descripción Breve	Requisitos sucios		
Descripción Detallada	Un usuario con rol de 'Escritor' o 'Administrador' puede marcar un requisito como sucio, indicando que si se modifica un requisito, se marcarán como sospechosos todos aquellos requisitos que estén relacionados con el requisito modificado, con el fin de identificarlos más fácilmente y someterlos a revisión. La "suciedad" se transmite en la dirección marcada por la asimetría de la relación, o en ambos sentidos si la relación en cuestión es simétrica.		
Fuente	Tutor	**Necesidad**	Opcional

4.2.1.1.3 Estadísticas e Informes

Identificador	00073-1	**Tipo**	Funcional
Descripción Breve	Estadísticas del proyecto		
Descripción Detallada	Un usuario asociado a un proyecto puede visualizar estadísticas asociadas al proyecto actual que reflejen datos relevantes del estado del proyecto.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00074-1	**Tipo**	Funcional
Descripción Breve	Datos mostrados en las estadísticas de un proyecto		
Descripción Detallada	Un usuario con permisos de 'Lector' sobre un proyecto puede ver en cualquier momento las estadísticas asociadas a un proyecto, donde se mostrarán los siguientes valores: - Paquetes del proyecto - 'Subpaquetes' medios por paquete - Profundidad máxima en la jerarquía de paquetes - Número total de requisitos - Número medio de requisitos por paquete - Número total de requisitos por tipo		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00075-1	**Tipo**	Funcional
Descripción Breve	Informes generados de un proyecto		
Descripción Detallada	Un usuario con rol 'Lector' sobre un proyecto puede generar informes respecto al proyecto actual. Los informes que pueden generar respecto al proyecto, son los siguientes: - Informe detallado de requisitos: consiste en un listado detallado de los requisitos del proyecto. - Tabla de requisitos: Consiste en un listado de los requisitos mostrado de manera tabular - Informe de relaciones: consiste en una matriz con las relaciones entre los requisitos del proyecto.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00076-1	**Tipo**	Funcional
Descripción Breve	Informe detallado de requisitos		
Descripción Detallada	A la hora de generar un informe de requisitos, el usuario puede elegir diferentes opciones para generar el informe a su gusto. En primer lugar, puede elegir los paquetes cuyos requisitos desea que aparezcan en el informe. En el informe se mostrarán cada uno de los requisitos de los paquetes seleccionados. Se seleccionarán que atributos de los requisitos serán mostrados en el informe. No se muestran aquellas que no estén habilitadas para el tipo del requisito. Por último, el usuario elige el formato del informe. Puede generarlo en formato HTML, en formato DOC o en formato OpenDocument.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00077-1	**Tipo**	Funcional
Descripción Breve	Informe de relaciones		
Descripción Detallada	Al generar un informe de relaciones el usuario va a generar una tabla o matriz con las relaciones existentes entre los requisitos del proyecto. El usuario puede elegir los paquetes cuyos requisitos desea que aparezcan en el informe. Si elige más de dos paquetes, se enfrentan entre sí todos los requisitos de los paquetes seleccionados y se marcan las relaciones entre ellos. Si se eligen dos paquetes, se ofrece al usuario la posibilidad de enfrentar en la matriz los requisitos de uno de los paquetes contra los del otro. También puede no elegir esta opción, con lo que se enfrentan todos contra todos al igual que cuando se seleccionan más de dos paquetes.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00078-1	**Tipo**	Funcional
Descripción Breve	Histórico de informes		
Descripción Detallada	Existe un histórico de los informes generados. Para cada proyecto, un usuario 'Administrador' puede ver un listado con los informes que se han generado, su fecha de generación, el autor del informe, el tipo de informe y el formato en el que se generó.		
Fuente	Tutor	**Necesidad**	Opcional

Identificador	00079-1	**Tipo**	Funcional
Descripción Breve	Requisitos modificados desde la última sesión		
Descripción Detallada	Un usuario puede visualizar los requisitos que hayan sido modificados desde el comienzo de su última sesión ordenados por fecha de modificación.		
Fuente	Tutor	**Necesidad**	Opcional

4.2.1.2 Acceso al sistema

Identificador	00080-1	**Tipo**	Funcional
Descripción Breve	Creación de cuentas		
Descripción Detallada	Cualquier persona puede crear una cuenta propia para poder acceder al sistema.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00081-1	**Tipo**	Funcional
Descripción Breve	Nuevo usuario. Datos personales		
Descripción Detallada	A la hora de crear un nuevo usuario, deben introducirse los siguientes campos que definirán al nuevo usuario: - Nombre de usuario, el cual debe ser único en todo el sistema - Nombre - Apellidos - E-mail - Contraseña		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00082-1	**Tipo**	Funcional
Descripción Breve	Características de la contraseña		
Descripción Detallada	La contraseña deberá ser una cadena de texto en la que se permiten tanto los caracteres alfanuméricos distinguiendo mayúsculas y minúsculas, como los siguientes símbolos: .,;:-_=/¬#$%&/()!?'`´+][^¨. La longitud de la cadena deberá ser mayor o igual de 8 caracteres y menor o igual de 30		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00083-1	**Tipo**	Funcional
Descripción Breve	Características del identificador		
Descripción Detallada	La contraseña deberá ser una cadena de texto en la que se permiten únicamente caracteres alfanuméricos distinguiendo mayúsculas y minúsculas. La longitud de la cadena deberá ser mayor o igual de 4 caracteres y menor o igual de 30.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00084-1	**Tipo**	Funcional
Descripción Breve	Borrar cuenta de usuario		
Descripción Detallada	Un usuario puede en cualquier momento, darse de baja del sistema. En ese momento su cuenta pasa a estar deshabilitada de forma que no pueda acceder al sistema. Sin embargo, su información permanecerá en el sistema.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00085-1	**Tipo**	Funcional
Descripción Breve	Regeneración de contraseña		
Descripción Detallada	Un usuario puede solicitar la regeneración de su contraseña en caso de que se olvide y se le comunique vía email la nueva contraseña		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00086-1	**Tipo**	Funcional
Descripción Breve	Cambiar contraseña		
Descripción Detallada	Un usuario puede cambiar en cualquier momento la contraseña con la que accede al sistema. Para realizar dicho cambio, debe indicar la contraseña actual y luego se indica la nueva contraseña. La nueva contraseña debe introducirse dos veces para evitar errores durante el tecleado de la misma		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00087-1	**Tipo**	Funcional
Descripción Breve	Cambiar datos personales		
Descripción Detallada	Un usuario puede cambiar los datos personales asociados a su cuenta. El único campo que no será posible variar será el nombre de usuario que identifica de manera única al usuario en el sistema.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00088-1	**Tipo**	Funcional
Descripción Breve	Apertura simultánea de proyectos		
Descripción Detallada	Un usuario que inicie sesiones tendrá acceso simultáneo a todos los proyectos que tenga asociado y estén abiertos		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00089-1	**Tipo**	Funcional
Descripción Breve	Apertura de proyectos cerrados		
Descripción Detallada	Un usuario podrá solicitar la visualización de los proyectos que hayan sido cerrados.		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00090-1	**Tipo**	Funcional
Descripción Breve	Datos proporcionados en el acceso al sistema		
Descripción Detallada	Un usuario podrá acceder al sistema especificando los siguientes datos: - Identificador. - Contraseña.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00091-1	**Tipo**	Funcional
Descripción Breve	Cerrar sesión		
Descripción Detallada	Un usuario puede cerrar sesión. En ese momento todos sus datos asociados se perderán y volverá a la pantalla de acceso al sistema donde puede iniciar sesión con otra cuenta, sin necesidad de reiniciar la aplicación.		
Fuente	Tutor	**Necesidad**	Esencial

4.2.1.2.1 Sesiones

Identificador	00092-1	**Tipo**	Funcional
Descripción Breve	Sesiones de un usuario		
Descripción Detallada	Se almacenarán las diferentes sesiones que un usuario realiza en el sistema. Almacenando tanto la fecha de entrada como de salida		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00093-1	**Tipo**	Funcional
Descripción Breve	Histórico de sesiones		
Descripción Detallada	El administrador de un proyecto podrá ver el histórico de sesiones de los usuarios que tengan acceso a dicho proyecto		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00094-1	**Tipo**	Funcional
Descripción Breve	Visualización de sesiones concurrentes		
Descripción Detallada	Cualquier usuario podrá ver que usuarios con acceso a alguno de sus proyectos están conectados de manera actual		
Fuente	Tutor	**Necesidad**	Esencial

4.2.1.3 Métricas

4.2.1.3.1 Proceso de Evaluación

Identificador	00095-1	**Tipo**	Funcional
Descripción Breve	Evaluación de requisitos		
Descripción Detallada	Un usuario con role 'Escritor' o Administrador sobre un proyecto puede realizar la evaluación de la calidad de requisitos a partir de ciertos atributos cuantificables que posea la descripción de un requisito		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00096-1	**Tipo**	Funcional
Descripción Breve	Elección de métrica para la evaluación de un proyecto		
Descripción Detallada	A la hora de realizar una evaluación de un proyecto, se puede seleccionar entre una de las métricas asociadas al proyecto, utilizándose por defecto, la métrica asociada por defecto al proyecto.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00097-1	**Tipo**	Funcional
Descripción Breve	Uso de la descripción detallada para la evaluación		
Descripción Detallada	El usuario puede definir durante el proceso de evaluación si desea que de los requisitos se evalúen su descripción corta o bien, su descripción detallada.		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00098-1	**Tipo**	Funcional
Descripción Breve	Definición de filtros para el proceso de evaluación de requisitos		
Descripción Detallada	El usuario podrá definir ciertos filtros que determinen los requisitos que serán sometidos al proceso de evaluación de requisitos. Entre las características que podrá definir, serán las siguientes: - Paquetes del proyecto actual cuyos requisitos serán evaluados. - Usuarios que han creado o modificado (o ambas) requisitos. - Fechas en las que se han creado o modificado requisitos. - Tipos de los requisitos evaluados.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00099-1	**Tipo**	Funcional
Descripción Breve	Filtrado por paquetes de requisitos a evaluar		
Descripción Detallada	A la hora de realizar el proceso de evaluación, el usuario evaluador puede elegir qué paquetes de los del proyecto serán utilizados para realizar la evaluación.		
Fuente	Alumno	**Necesidad**	Deseable

Identificador	00100-1	**Tipo**	Funcional
Descripción Breve	Filtrado por usuarios para la evaluación de requisitos		
Descripción Detallada	El usuario evaluador puede definir aquellos usuarios responsables de los requisitos que queremos evaluar, para eso se definen: - Si se desea tener en cuenta que los usuarios sean creadores o modificadores de los requisitos, cualquiera de ellos o ambas. - Los usuarios por los cuales se desean filtrar		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00101-1	**Tipo**	Funcional
Descripción Breve	Filtrado por fechas para la evaluación de requisitos		
Descripción Detallada	El usuario evaluador puede filtrar por fecha los requisitos evaluados. Para ello se especifica una fecha de inicio y una fecha de fin, siendo evaluados todos los requisitos cuya fecha de creación o modificación, según sea definido por el usuario, estén contenidos en el intervalo especificado		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00102-1	**Tipo**	Funcional
Descripción Breve	Recomendaciones posteriores a la evaluación de requisitos		
Descripción Detallada	Una vez realizada la evaluación de los requisitos y mostrados los resultados, se realizarán ciertas recomendaciones siguiendo la 'visión del consejero' para la mejora de los requisitos a partir de la métrica utilizada		
Fuente	Tutor	**Necesidad**	Opcional

Identificador	00103-1	**Tipo**	Funcional
Descripción Breve	Obtención de valoración global de la evaluación		
Descripción Detallada	Una vez evaluado los requisitos, en caso de que el evaluador tenga permisos de 'Administrador' sobre el proyecto, se obtendrá un conjunto de valoraciones globales para el mismo.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00104-1	**Tipo**	Funcional
Descripción Breve	Obtención de valoración individual de cada requisito		
Descripción Detallada	Una vez evaluado los requisitos se obtendrá una valoración global para cada requisito que haya sido evaluado.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00105-1	**Tipo**	Funcional
Descripción Breve	Detalles de la evaluación individual de un requisito		
Descripción Detallada	El usuario puede mostrar los resultados de la evaluación individual, donde se mostrarán para cada indicador el valor nominal obtenido en el mismo, así como el valor número del recuento asociado a dicho indicador.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00106-1	**Tipo**	Funcional
Descripción Breve	Generación de informes de evaluación		
Descripción Detallada	El usuario puede generar un informe asociado a la evaluación de un determinado conjunto de requisitos.		
Fuente	Alumno	**Necesidad**	Deseable

Identificador	00107-1	**Tipo**	Funcional
Descripción Breve	Datos mostrados en el informe de evaluación		
Descripción Detallada	El informe contendrá además de los resultados que se han obtenido en la evaluación, se mostrarán las características con las que se ha realizado la evaluación: - Datos de la métrica utilizada. - Datos de los indicadores utilizados - Datos del dominio utilizado. - Datos de los filtros aplicados sobre el conjunto de requisitos		
Fuente	Alumno	**Necesidad**	Deseable

Identificador	00108-1	**Tipo**	Funcional
Descripción Breve	Exportación de informes de evaluación.		
Descripción Detallada	El usuario puede exportar los informes generados por la aplicación a un fichero en almacenamiento secundario en los siguientes formatos: - Formato PDF - Formato DOC - Formato XLS		
Fuente	Tutor	**Necesidad**	Esencial

4.2.1.3.2 Atributos de una métrica

Identificador	00109-1	**Tipo**	Funcional
Descripción Breve	Definición de métricas		
Descripción Detallada	Un usuario puede definir las características de los atributos cuantificables que serán aplicadas para determinar la calidad de un requisito. El conjunto de dichas características para cada uno de los atributos se definirán como una métrica.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00110-1	**Tipo**	Funcional
Descripción Breve	Atributos asociados a una métrica		
Descripción Detallada	Una métrica tiene un conjunto de atributos asociados, que son los siguientes: -Nombre: identificativo de cada métrica y sirve para que sea fácilmente reconocible por los usuarios. -Indicadores asociados: Conjunto de características de los atributos cuantificables que serán aplicadas para determinar la calidad de un requisito. - Fecha de modificación: Fecha en la que se ha producido la última modificación sobre la métrica asociada - Modificador: Usuario último que ha realizado una modificación sobre la métrica - Usuarios asociados: Conjunto de usuarios asociados a la métrica con uno de los roles asociados - Valores nominales: Conjunto de valores nominales para los diferentes índices de calidad. - Valores agregados: Conjunto de valores agregados para los diferentes niveles de calidad. - Función de agregación asociada: Tipo de función de agregación utilizada.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00111-1	**Tipo**	Funcional
Descripción Breve	Atributos automáticos y no modificables asociados a una métrica		
Descripción Detallada	Los atributos que se muestran a continuación no podrán ser modificados de manera manual: - Fecha de modificación. - Modificador.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00112-1	**Tipo**	Funcional
Descripción Breve	Roles sobre una métrica		
Descripción Detallada	Una métrica tendrá un conjunto de usuarios asociados que podrán, en función de su rol sobre la métrica, modificar o utilizar dicha métrica. Cada usuario asociado tendrá uno de los siguientes roles jerárquicos: - Usador: El usuario puede hacer uso de la métrica y evaluar con dicha métrica. - Definidor: El usuario puede modificar la métrica Los roles son jerárquicos, de modo que todas las acciones que estén disponibles para un rol inferior estarán siempre disponibles para un rol superior.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00113-1	**Tipo**	Funcional
Descripción Breve	Añadir un indicador a una métrica		
Descripción Detallada	Un usuario con permisos de 'Definidor' puede añadir un indicador a una métrica, especificando los siguientes valores: - Recuento utilizado: Se deberá elegir entre uno de los recuentos definidos dentro de la aplicación. - Predominante: Se deberá indicar si el indicador actúa como predominante o no - Peso: Se deberá indicar el peso de indicador mediante un valor numérico - Intervalos: Se deberán definir un conjunto de intervalos, los cuales definirán el valor de dicha métrica. El número de intervalos puede ser 3 ó 5, en función del tipo de recuento utilizado, por tanto deberá definirse 2 ó 4 límites entre dichos intervalos, que consistiría en valores enteros en orden creciente.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00114-1	**Tipo**	Funcional
Descripción Breve	Modificar un indicador asociado a una métrica		
Descripción Detallada	Un usuario, con permisos de 'Definidor' sobre una métrica puede modificar un indicador asociado a una métrica, pudiendo modificar todos los valores especificados en el momento de la creación.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00115-1	**Tipo**	Funcional
Descripción Breve	Eliminar un indicador asociado a una métrica		
Descripción Detallada	Un usuario, con permisos de 'Definidor' sobre una métrica puede eliminar un indicador asociado a una métrica, seleccionado el indicador a eliminar		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00116-1	**Tipo**	Funcional
Descripción Breve	Añadir un usuario a una métrica		
Descripción Detallada	Un usuario, con permisos de 'Definidor' sobre una métrica puede añadir otro usuario a la métrica para que pueda hacer uso de la misma, especificando el 'nombre de usuario' que identifica unívocamente al usuario.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00117-1	**Tipo**	Funcional
Descripción Breve	Eliminar un usuario a una métrica		
Descripción Detallada	Un usuario, con permisos de 'Definidor' sobre una métrica puede eliminar a otro usuario asociado a la métrica, siempre que su rol no sea el de ‘Definidor’		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00118-1	**Tipo**	Funcional
Descripción Breve	Modificar los permisos de un usuario sobre una métrica		
Descripción Detallada	Un usuario, con permisos de 'Definidor' sobre una métrica puede modificar los permisos de cualquier otro usuario asociado a la métrica.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00119-1	**Tipo**	Funcional
Descripción Breve	Definición de la función de agregación		
Descripción Detallada	Una métrica contiene la función de agregación que se utiliza en el proceso de evaluación. Los valores que puede tomar, son los siguientes: - Promedio - Mínimo - Mixto		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00120-1	**Tipo**	Funcional
Descripción Breve	Modificación de una métrica		
Descripción Detallada	Un usuario con permisos de 'Definidor' sobre una métrica, puede modificar cualquiera de los atributos 'modificables' de una métrica		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00121-1	**Tipo**	Funcional
Descripción Breve	Importación y exportación de métricas		
Descripción Detallada	Un usuario podrá exportar una métrica a un fichero XML siempre que tenga rol 'Definidor' sobre la misma. Del mismo modo, siguiendo el mismo formato podrá importar métricas en el sistema. Un usuario tiene automáticamente el rol de 'Definidor' sobre la métrica que importa.		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00122-1	**Tipo**	Funcional
Descripción Breve	Eliminación de métricas		
Descripción Detallada	Un usuario podrá determinar la eliminación de una métrica. En caso de que existan más usuarios con permisos de 'Definidor' sobre esa métrica el único efecto es que el usuario borrar 'su acceso' a la misma. En caso de que no existan más usuarios o estos sean 'Usadores' la métrica se eliminará, pero sus vinculaciones quedarán activas.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00123-1	**Tipo**	Funcional
Descripción Breve	Duplicación de métricas		
Descripción Detallada	Un usuario podrá determinar la duplicación de una métrica. En ese caso se creará una copia con los mismos indicadores y sus valores asociados. El único cambio reseñable se verá en el nombre que figurará 'Copia de' antes del nombre de la métrica original. El usuario que realiza la duplicación pasa a ser automáticamente 'Definidor' de la misma		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00124-1	**Tipo**	Funcional
Descripción Breve	Invalidación de recuentos debido a la modificación de una métrica		
Descripción Detallada	Cuando se realice un cambio sobre una métrica, en función del cambio, puede ser necesario invalidar todos los requisitos de todos los proyectos que tengan esa métrica establecida como 'por defecto'. Se invalidará si bien, se añade un nuevo indicador que esté asociado a una lista de palabras o si se modifica una lista de palabras que este asociado a un indicador activo en la métrica..		
Fuente	Tutor	**Necesidad**	Esencial

4.2.1.3.3 Recuentos asociados a un indicador

Identificador	00125-1	**Tipo**	Funcional
Descripción Breve	Recuentos por defecto		
Descripción Detallada	Existirá un conjunto de recuentos por defecto cuyos datos son inmutables y no pueden ser eliminados del sistema. Dichos recuentos se encuentran descritos en el Anexo III. Recuentos establecidos por defecto		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00126-1	**Tipo**	Funcional
Descripción Breve	Añadir un recuento		
Descripción Detallada	Cualquier usuario puede crear un nuevo recuento al sistema que deberá estar asociado a una de la lista de palabras a la que tenga acceso. Además deberá proporcionar los siguientes campos: - Nombre: Nombre del recuento - Descripción: Descripción del recuento - Función Escalonada : Función escalonada del recuento - Categoría: Categoría del recuento.		
Fuente	Alumno	**Necesidad**	Deseable

Identificador	00127-1	**Tipo**	Funcional
Descripción Breve	Categorías de un recuento		
Descripción Detallada	Las categorías que pueden ser asociados a un recuento son las siguientes: - Léxico - Morfológico - Analítico - Relacional		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00128-1	**Tipo**	Funcional
Descripción Breve	Función escalonada de un recuento		
Descripción Detallada	Las funciones escalonadas que pueden estar asociadas a un recuento, son los siguientes: - Creciente - Decreciente - Cóncavo - Convexo		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00129-1	**Tipo**	Funcional
Descripción Breve	Borrar un recuento		
Descripción Detallada	Un usuario puede borrar un recuento del sistema siempre que haya sido creado por él y no pertenezca al grupo de recuentos por defecto de la aplicación.		
Fuente	Alumno	**Necesidad**	Deseable

Identificador	00130-1	**Tipo**	Funcional
Descripción Breve	Modificar un recuento		
Descripción Detallada	Un usuario puede modificar un recuento del sistema siempre que haya sido creado por él y no pertenezca al grupo de recuentos por defecto de la aplicación.		
Fuente	Alumno	**Necesidad**	Deseable

Identificador	00131-1	**Tipo**	Funcional
Descripción Breve	Limitación de recuentos asociados a una lista de palabras		
Descripción Detallada	Dada la utilización del módulo CAKE, el número de recuentos asociados a listas de palabras que pueden estar activados simultáneamente o asociados a una métrica es de 9		
Fuente	Tutor	**Necesidad**	Esencial

4.2.1.4 Dominios y Listas de palabras

4.2.1.4.1 Dominio

Identificador	00132-1	**Tipo**	Funcional
Descripción Breve	Definición de Dominios		
Descripción Detallada	El usuario puede definir nuevos dominios que sean relevantes de cara a los requisitos asociados a un proyecto, entendiendo como dominio aquellos términos que tengan una relevancia especial en el ámbito del proyecto en cuestión.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00133-1	**Tipo**	Funcional
Descripción Breve	Atributos de un dominio		
Descripción Detallada	Un dominio contiene un conjunto de atributos asociados, que son los siguientes: - Nombre del dominio - Términos del dominio - Fecha de última modificación - Último modificador. - Usuarios asociados al dominio		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00134-1	**Tipo**	Funcional
Descripción Breve	Roles sobre un dominio		
Descripción Detallada	Un dominio tendrá un conjunto de usuarios asociados que podrán, en función de su rol sobre el mismo, modificar o utilizar dicho dominio. Cada usuario asociado tendrá uno de los siguientes roles jerárquicos: - Usador: El usuario puede hacer uso del dominio y asociarlo a algún proyecto. - Definidor: El usuario puede modificar el dominio Los roles son jerárquicos, de modo que todas las acciones que estén disponibles para un rol inferior estarán siempre disponibles para un rol superior.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00135-1	**Tipo**	Funcional
Descripción Breve	Términos asociados a un dominio		
Descripción Detallada	Un usuario podrá añadir, modificar o borrar términos asociados a un dominio, estando cada uno de ellos compuesto de los siguientes campos: - nombre: Nombre del término que deberá ser único dentro del dominio asociado., siendo una cadena de caracteres entre 1 y 100 caracteres. - tipo: Indica si se trata de un sustantivo o un verbo. - variación de número: Bandera que determina si se permite la variación del número de dicho término. (Sólo para sustantivos) - variación de género: Término que especifica, en caso de que proceda, el término una vez aplicada la variación de género. (Sólo para sustantivos)		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00136-1	**Tipo**	Funcional
Descripción Breve	Añadir un usuario a una dominio		
Descripción Detallada	Un usuario, con permisos de 'Definidor' sobre un dominio puede añadir otro usuario al dominio para que pueda hacer uso del mismo, especificando el 'nombre de usuario' que identifica unívocamente al usuario.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00137-1	**Tipo**	Funcional
Descripción Breve	Eliminar un usuario a un dominio		
Descripción Detallada	Un usuario, con permisos de 'Definidor' sobre un dominio puede eliminar a otro usuario asociado al dominio.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00138-1	**Tipo**	Funcional
Descripción Breve	Modificar los permisos de un usuario sobre un dominio		
Descripción Detallada	Un usuario, con permisos de 'Definidor' sobre un dominio puede modificar los permisos de cualquier otro usuario asociado al dominio.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00139-1	**Tipo**	Funcional
Descripción Breve	Importación/Exportación de Dominios		
Descripción Detallada	Un usuario puede importar y exportar un dominio mediante un fichero XML que cumpla un determinador XML Schema		
Fuente	Alumno	**Necesidad**	Deseable

Identificador	00140-1	**Tipo**	Funcional
Descripción Breve	Eliminación de Dominios		
Descripción Detallada	Un usuario podrá determinar la eliminación de un dominio. En caso de que existan más usuarios con permisos de 'Definidor' sobre ese dominio el único efecto es que el usuario borrar 'su acceso' al mismo. En caso de que no existan más usuarios con permisos de 'Definidor' el dominio se eliminará completamente del sistema.		
Fuente	Tutor	**Necesidad**	Esencial

4.2.1.4.2 Lista de palabras

Identificador	00141-1	**Tipo**	Funcional
Descripción Breve	Definición de listas de palabras		
Descripción Detallada	El usuario puede definir un número limitado de listas de palabras que podrán estar asociados a un indicador con el fin de determinar la presencia en la descripción de los requisitos de los términos asociados a una determinada lista.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00142-1	**Tipo**	Funcional
Descripción Breve	Importación/Exportación de Listas de Palabras		
Descripción Detallada	Se puede importar y exportar una lista de palabras asociados a un recuento mediante un fichero XML que cumpla un determinador XML Schema		
Fuente	Alumno	**Necesidad**	Deseable

Identificador	00143-1	**Tipo**	Funcional
Descripción Breve	Definición de Términos Especiales		
Descripción Detallada	Cualquier usuario podrá modificar, añadir o borrar términos asociados a las listas de términos especiales. Para ello, cada término se compondrá de los siguientes campos: - nombre: Cadena de caracteres entre 1 y 100 caracteres.		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00144-1	**Tipo**	Funcional
Descripción Breve	Atributos de una lista de palabras		
Descripción Detallada	Una lista de palabras contiene un conjunto de atributos asociados, que son los siguientes: - Nombre de la lista de palabras - Términos de la lista de palabras - Fecha de última modificación - Último modificador. - Usuarios asociados a la lista de palabras		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00145-1	**Tipo**	Funcional
Descripción Breve	Roles sobre una lista de palabras		
Descripción Detallada	Una lista de palabras tendrá un conjunto de usuarios asociados que podrán, en función de su rol sobre la lista, modificar o utilizar dicha lista como recuento. Cada usuario asociado tendrá uno de los siguientes roles jerárquicos: - Usador: El usuario puede hacer uso de la lista de palabras y asociarlo a un indicador. - Definidor: El usuario puede modificar la lista de palabras. Los roles son jerárquicos, de modo que todas las acciones que estén disponibles para un rol inferior estarán siempre disponibles para un rol superior.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00146-1	**Tipo**	Funcional
Descripción Breve	Añadir un usuario a una lista de palabras		
Descripción Detallada	Un usuario, con permisos de 'Definidor' sobre una lista de palabras puede añadir otro usuario a la lista de palabras para que pueda hacer uso de la misma, especificando el 'nombre de usuario' que identifica unívocamente al usuario.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00147-1	**Tipo**	Funcional
Descripción Breve	Eliminar un usuario a una lista de palabras		
Descripción Detallada	Un usuario, con permisos de 'Definidor' sobre una lista de palabras puede eliminar a otro usuario asociado a la lista de palabras cuyo rol no sea el de Definidor.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00148-1	**Tipo**	Funcional
Descripción Breve	Modificar los permisos de un usuario sobre una lista de palabras		
Descripción Detallada	Un usuario, con permisos de 'Definidor' sobre una lista de palabras puede modificar los permisos de cualquier otro usuario asociado a la lista de palabras.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00149-1	**Tipo**	Funcional
Descripción Breve	Eliminación de Listas de palabras		
Descripción Detallada	Un usuario podrá determinar la eliminación de una lista de palabras. En caso de que existan más usuarios con permisos de 'Definidor' sobre esa lista el único efecto es que el usuario borrar 'su acceso' al mismo. En caso de que no existan más usuarios con permisos de 'Definidor' la lista de palabras se eliminará completamente del sistema.		
Fuente	Tutor	**Necesidad**	Esencial

4.2.1.5 Experimentos

4.2.1.5.1 Proyectos de entrenamiento

Identificador	00150-1	**Tipo**	Funcional
Descripción Breve	Conjunto de proyectos de entrenamiento		
Descripción Detallada	Cada usuario del sistema tiene un conjunto de proyectos de entrenamiento, los cuales son utilizados para el proceso de generación de métricas. El conjunto de proyectos de entrenamiento se almacenan en la base de datos asociado al sistema		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00151-1	**Tipo**	Funcional
Descripción Breve	Atributos de un proyecto de entrenamiento		
Descripción Detallada	Un proyecto de entrenamiento tiene un conjunto de atributos asociados: - Nombre: Nombre del proyecto. - Grupo: Nombre del grupo del cual forma parte el proyecto. - Medida externa de calidad: Medida de calidad del proyecto de entrenamiento que toma un valor numérico con decimales sin restricción de intervalos. - Dominio asociado: Dominio que se encuentra asociado al proyecto. - Creador: Creador o importador del proyecto. - Fecha de Creación: Fecha de creación del proyecto		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00152-1	**Tipo**	Funcional
Descripción Breve	Importación y exportación de proyectos de entrenamiento		
Descripción Detallada	Se podrán importar nuevos requisitos y proyectos de entrenamiento a partir de las siguientes fuentes: - Base de datos Access, según el formato especificado en II.1.3.4 Formato Microsoft Access 2003 y 2007 - Fichero Excel 2003 y 2007, según el formato especificado en II.1.3.2 Formato Microsoft Excel 2003 y 2007 - Fichero Word 2003 y 2007 previamente etiquetado, según el formato especificado en II.1.3.3 Formato Microsoft Word 2003 y 2007 - Ficheros de texto plano, según el formato especificado en II.1.3.5 Formato Fichero de texto plano - Ficheros XML que cumplen un XML Schema determinado Los proyectos importados pasarán a formar parte del conjunto de		

	entrenamiento del usuario que realiza la importación de los mismos. Del mismo modo, se podrán exportar los proyectos de entrenamiento a un fichero XML que cumpla con un XML Schema determinado.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00153-1	**Tipo**	Funcional
Descripción Breve	Selección de proyectos de entrenamiento		
Descripción Detallada	El usuario puede elegir cuales de los proyectos de entrenamiento se utilizan como parte del experimento para la obtención de nuevos individuos, debiendo escoger como mínimo dos proyectos. Del mismo modo el usuario puede seleccionar de manera automática, todos los proyectos o todos los proyectos de un determinado grupo.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00154-1	**Tipo**	Funcional
Descripción Breve	Edición de proyectos de entrenamiento		
Descripción Detallada	Un usuario puede editar las características o los requisitos asociados a un proyecto de entrenamiento si forma parte de su conjunto de entrenamiento. Los cambios serán visibles para todos los usuarios de dicho proyecto de entrenamiento.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00155-1	**Tipo**	Funcional
Descripción Breve	Borrar proyectos de entrenamiento		
Descripción Detallada	Un usuario puede borrar proyectos de entrenamiento que formen parte del conjunto de entrenamiento del sistema. Sólo desaparecerán del sistema si no forman parte del conjunto de entrenamiento de ningún otro usuario.		
Fuente	Tutor	**Necesidad**	Deseable

4.2.1.5.2 Proceso de generación de métricas

Identificador	00156-1	**Tipo**	Funcional
Descripción Breve	Proceso de generación de métricas		
Descripción Detallada	El sistema permite la obtención de nuevas métricas a partir de un proceso de generación basado en algoritmos genéticos.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00157-1	**Tipo**	Funcional
Descripción Breve	Modo de realización del proceso de generación de métricas		
Descripción Detallada	El proceso de entrenamiento y generación de nuevas métricas se realizará siguiendo las directrices del artículo 'Afinamiento de métricas de calidad en los requisitos'.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00158-1	**Tipo**	Funcional
Descripción Breve	Definición de parámetros del proceso de generación de métricas		
Descripción Detallada	Para la realización de un proceso de generación de métricas se deben definir los siguientes parámetros: - Tamaño de la población (Número entero entre 1 y 100000) - Número de generaciones. (Número entero entre 1 y 100000) - Tamaño del torneo (Número entero entre 1 y 100) - Número de generaciones recientes (Número entero entre 1 y 1000) - Umbral de parada (Valor decimal con 5 cifras significativas tomando como valor mínimo 0) - Porcentaje de selección (Porcentaje entre 1 y 100) - Probabilidad de mutación (Porcentaje entre 1 y 100)		

	- Medida Global de Calidad (Q,Q',Q") - Medida Individual de Calidad (Promedio, Mínimo y Mixto) - Población inicial: Lista de métricas que se utilizan inicialmente en el proceso. - Conjunto de proyectos de entrenamiento a utilizar - Valores máximo y mínimo de la medida externa de calidad. Establecerá aquellos valores que actuarán como extremos de la medida externa de calidad con el fin de normalizarlos en la misma escala que la medida global proporcionada por las métricas.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00159-1	**Tipo**	Funcional
Descripción Breve	Definición de los valores nominales y de agregación		
Descripción Detallada	Un usuario debe especificar los valores numéricos asociados a los valores nominales de calidad de un requisito, (Malo, Medio y Bueno), así como los valores que delimitan los conjuntos de la 'normalización' de los valores una vez aplicada la función de agregación		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00160-1	**Tipo**	Funcional
Descripción Breve	Selección de recuentos utilizados en el proceso de generación de métricas		
Descripción Detallada	El usuario debe indicar que recuentos de características cuantificables se quiere usar en la obtención de métricas. Dichos recuentos serán la base de los indicadores que contendrán cada una de las métricas generadas por la aplicación		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00161-1	**Tipo**	Funcional
Descripción Breve	Almacenamiento de métricas generadas durante el proceso de entrenamiento		
Descripción Detallada	Un usuario puede almacenar cualquier métrica generada durante el proceso de entrenamiento no siendo necesario esperar a que finalice dicho proceso. Dichas métricas pasan a formar parte del 'listado de métricas del usuario' y el usuario es 'Definidor' de dicha métrica.		
Fuente	Alumno	**Necesidad**	Deseable

Identificador	00162-1	**Tipo**	Funcional
Descripción Breve	Limitación de la mutación de un individuo		
Descripción Detallada	Durante el proceso de entrenamiento se podrán congelar el valor de ciertos indicadores y las modificaciones se limitarán a los restantes. La congelación hará referencia a los siguientes valores asociados al indicador: - Intervalos especificados - Predominante o no - Peso del indicador.		
Fuente	Tutor	**Necesidad**	Opcional

Identificador	00163-1	**Tipo**	Funcional
Descripción Breve	Detener momentáneamente el proceso de entrenamiento		
Descripción Detallada	Un usuario podrá detener momentáneamente el proceso de entrenamiento, pudiendo reanudarlo desde la misma interfaz, siempre que no salga de la aplicación ni abandone el proceso.		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00164-1	**Tipo**	Funcional
Descripción Breve	Cancelación del proceso de evaluación		
Descripción Detallada	Un usuario podrá detener el proceso de evaluación cancelándolo, de forma que dicho proceso no podrá volver a ser reanudado		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00165-1	**Tipo**	Funcional
Descripción Breve	Selección de un experimento anterior		
Descripción Detallada	Un usuario puede cargar los datos de configuración de un experimento anterior, debido a que se almacenan en la base de datos.		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00166-1	**Tipo**	Funcional
Descripción Breve	Compartición de experimentos		
Descripción Detallada	Un usuario puede compartir un experimento anterior que haya realizado con otro usuario. Para ello debe proporcionar el nombre de usuario identificativo del usuario con el que quiere compartir el experimento.		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00167-1	**Tipo**	Funcional
Descripción Breve	Importación y exportación de la población inicial y parámetros de configuración.		
Descripción Detallada	Un usuario puede importar y exportar los parámetros de configuración y la población inicial utilizada en un proceso, de forma que dicho experimento pueda ser replicado variando alguna de las características.		
Fuente	Tutor	**Necesidad**	Opcional

Identificador	00168-1	**Tipo**	Funcional
Descripción Breve	Generación de informes del experimento		
Descripción Detallada	Una vez finalizado el proceso de entrenamiento se genera un informe que contempla los principales datos asociados al experimento		
Fuente	Alumno	**Necesidad**	Opcional

4.2.2 Requisitos No funcionales

Identificador	00169-1	**Tipo**	No Funcional
Descripción Breve	Ubicación de la base de datos		
Descripción Detallada	La base de datos que contiene los datos de la aplicación podrá estar en una ubicación remota		
Fuente	Alumno	**Necesidad**	Deseable

Identificador	00170-1	**Tipo**	No Funcional
Descripción Breve	Conexión simultánea a la fuente de datos		
Descripción Detallada	Varios usuarios podrán simultáneamente estar trabajando sobre los mismos proyectos, siendo responsabilidad de la aplicación actualizar los cambios que se produzcan remotamente cada cierto tiempo.		
Fuente	Alumno	**Necesidad**	Deseable

Identificador	00171-1	**Tipo**	No Funcional
Descripción Breve	Hilo de preevaluación		
Descripción Detallada	El sistema se encarga de ir recalculando en un segundo plano los valores numéricos asociados a los atributos cuantificables de las descripciones de los requisitos con el fin de que el usuario no tenga que perder tiempo en largas esperas.		
Fuente	Tutor	**Necesidad**	Deseable

Identificador	00172-1	**Tipo**	No Funcional
Descripción Breve	Lenguaje de programación		
Descripción Detallada	La aplicación será desarrollada utilizando la tecnología Microsoft .NET, y más concretamente el lenguaje C#.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00173-1	**Tipo**	No Funcional
Descripción Breve	Localización de la implementación		
Descripción Detallada	La implementación facilitará la localización de la aplicación a diferentes idiomas mediante el uso de archivos de recursos.		
Fuente	Alumno	**Necesidad**	Deseable

Identificador	00174-1	**Tipo**	No Funcional
Descripción Breve	Base de datos soportada		
Descripción Detallada	El sistema soportará al menos Microsoft SQL Server como base de datos en su versión de 2005 o posterior.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00175-1	**Tipo**	No Funcional
Descripción Breve	Requisitos Hardware		
Descripción Detallada	La aplicación no requerirá capacidades más altas de rendimiento de procesador y de memoria de las que posee cualquier PC común del mercado actual.		
Fuente	Alumno	**Necesidad**	Deseable

Identificador	00176-1	**Tipo**	No Funcional
Descripción Breve	Almacenamiento de la configuración del sistema		
Descripción Detallada	Existe un fichero de configuración de la aplicación donde estarán almacenados las diferentes conexiones de un usuario, así como el idioma de uso de la aplicación.		
Fuente	Alumno	**Necesidad**	Deseable

Identificador	00177-1	**Tipo**	No Funcional
Descripción Breve	Requisitos Software. Informes		
Descripción Detallada	Para la generación de informes en formatos correspondientes a la suite ofimática Microsoft Office, es necesario que ésta se encuentre instalada en el sistema.		
Fuente	Alumno	**Necesidad**	Esencial

Identificador	00178-1	**Tipo**	No Funcional
Descripción Breve	Requisitos Software. Importación de proyectos		
Descripción Detallada	Para la importación de proyectos a partir de un fichero con formato Microsoft Word deberá encontrarse dicho programa instalado en el sistema.[2]		
Fuente	Alumno	**Necesidad**	Esencial

[2] Esto es así porque es necesario utilizar la API de Word. Por el contrario, para importar proyectos a partir de ficheros con formato Microsoft Excel y Microsoft Access no es necesario tener instaladas estas aplicaciones, ya que se usan conectores ODBC

Identificador	00179-1	**Tipo**	No Funcional
Descripción Breve	Compatibilidad con ReqStudio		
Descripción Detallada	El sistema debe permitir trabajar con los proyectos definidos en la herramienta ReqStudio hasta la versión 2.1 del mismo. Para ello se realizará una importación y conversión de los proyecto definidos en esta herramienta a ReqStudio Plus.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00180-1	**Tipo**	No Funcional
Descripción Breve	Almacenamiento de contraseñas		
Descripción Detallada	Las contraseñas de los usuarios no se almacenan, sino que, por seguridad, se almacena el resultado de aplicar una función resumen SHA-256 sobre las mismas.		
Fuente	Tutor	**Necesidad**	Esencial

Identificador	00181-1	**Tipo**	No Funcional
Descripción Breve	Seguridad de los datos		
Descripción Detallada	El cifrado de los datos durante su transmisión se delega sobre la configuración del sistema gestor de la base de datos.		
Fuente	Tutor	**Necesidad**	Esencial

5 Diseño Arquitectónico

El método de diseño elegido responde a la filosofía *top-down*, de modo que se parte de una visión general del sistema para, a continuación, pasar a detallar cada uno de los componentes que lo forman. Además, se ha elegido una arquitectura por capas, en la que cada capa estará formada por un número variable de componentes. Como queda definido en el siguiente punto de este apartado, el número de capas elegidas es de cuatro.

Los componentes de cada capa podrán contener, a su vez, nuevos componentes. Por último, en el nivel más bajo de descomposición, aparecen, dentro de cada componente, las diferentes clases que constituyen la aplicación.

5.1 Descomposición arquitectónica

Antes de mostrar la especificación de cada uno de los componentes que conforman la aplicación, es necesario explicar y justificar la descripción arquitectónica escogida. En este sistema la descripción es una descomposición en cuatro capas que busca minimizar las dependencias existentes entre cada una de las capas. De este modo se busca maximizar las opciones de mantenibilidad de la aplicación, así como que sea posible la sustitución de componentes de forma que el resto del sistema no se vea afectado o, al menos, la modificación no afecte gravemente con el fin de facilitar la tarea a los encargados del mantenimiento de la misma. El sistema sigue la descomposición que se muestra en la siguiente ilustración:

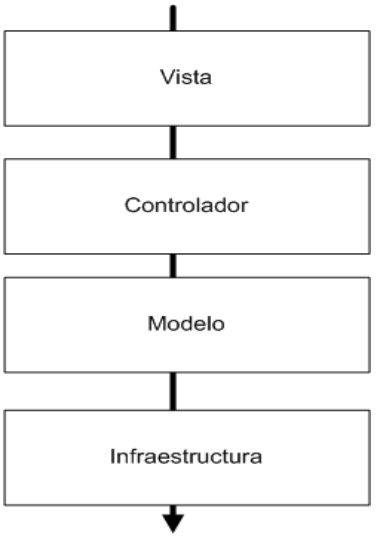

Ilustración 15: Esquema de la arquitectura del sistema

Las capas en las que se ha dividido el sistema son, como puede verse en la figura anterior, las siguientes:

- **Vista**. Los componentes que formen esta capa serán todos aquellos con los que puedan interactuar los usuarios. Recibirá peticiones de los usuarios y les mostrará resultados. Contiene toda la representación gráfica de la aplicación. Su punto de comunicación con el resto del sistema es a través de un controlador

- **Controlador.** El control es un componente que se encarga de recibir las peticiones por parte de la Vista y se comunica con el modelo con el fin de satisfacer las peticiones requeridas. Además, se encarga de realizar la comprobación de que el correcto formato de las peticiones y sus argumentos, en cuanto a longitudes, tipo de datos y similar.

- **Modelo.** Contiene el modelo de datos de la aplicación y la lógica asociada a los mismos, se encarga de la cumplir con la funcionalidad de la aplicación.

- **Infraestructura.** Es una capa que se encarga de la comunicación con entidades externas como puede ser la base de datos del sistema o el módulo de obtención de recuentos asociados a los requisitos. Del mismo modo se encarga de la exportación de los diferentes objetos.

Cabe destacar que estas cuatro capas corren en el cliente, mientras que la única parte del sistema que reside o puede residir en un lugar externo al cliente es el sistema gestor de base de datos, tal y como se mostró en 3.5 Entorno operacional.

Una vez explicados el propósito y objetivo de cada una de las clases, es necesario mostrar la descomposición real en componentes, tal y como se muestra en la Ilustración 16:

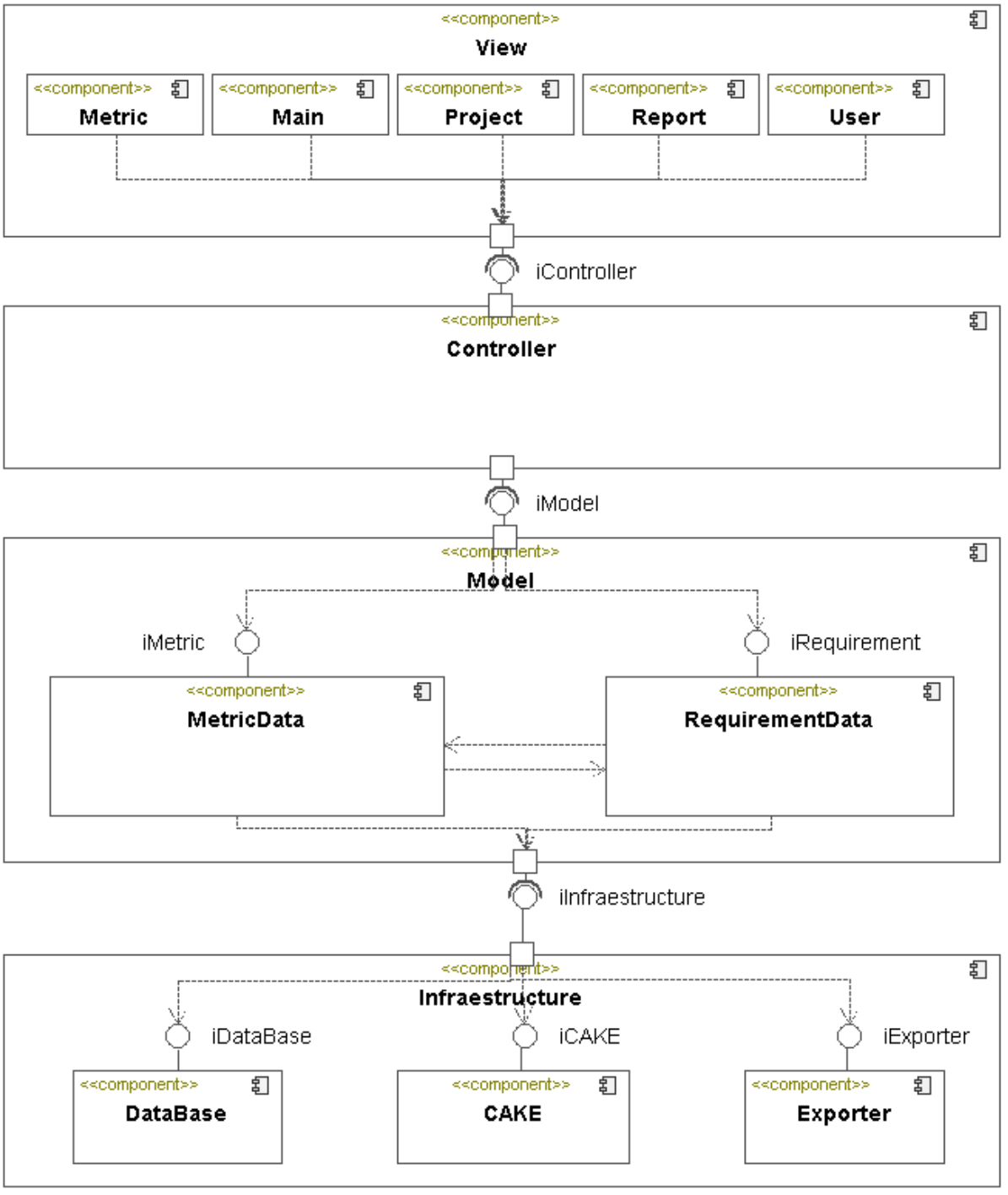

Ilustración 16: Diagrama de componentes de la aplicación

En la descomposición mostrada en la Ilustración 16, se puede ver claramente como existe un componente para cada una de las capas explicadas anteriormente. Del mismo modo, algunos de esos componentes se siguen descomponiendo a su vez en otros componentes subordinados, como es el caso de la Vista (componente *View*), el modelo (componente *Model*) o la infraestructura (componente *Infraestructure*):

El componente a *View* a su vez dispone de un conjunto de cinco componentes que se encargan de distribuir las clases que se encargan de diferentes ventanas y diálogos que componen la interfaz gráfica en función del propósito y finalidad de las mismas.

El componente asociado al modelo, a su vez, se descompone en dos componentes. El primero de ellos se encarga de los datos asociados a la evaluación de requisitos (*MetricData*), mientras que el segundo se encarga de la gestión de requisitos (*RequirementData*)

La capa de la infraestructura, se divide en tres componentes cada uno de ellos se ocupa de la interacción con un "entorno diferente". Por un lado, estaría el componente *DataBase* se encarga de la persistencia del modelo de datos de la aplicación a través de una base de datos.

La conexión con la base de datos y la aplicación se realiza a través de un conector ODBC, el cual será configurado a través de la propia aplicación. Esta decisión se ha visto motivada con el fin de facilitar la interoperabilidad de la aplicación con nuevas bases de datos, en lugar de utilizar un conector propio para cada sistema gestor de base de datos, favoreciendo de ese modo la mantenibilidad y la interoperabilidad sobre el pequeño aumento de eficiencia de utilizar un conector propio para cada base de datos.

El componente *CAKE*, se encarga de la relación del sistema con el sistema encargado de la obtención de recuentos de los indicadores sobre las descripciones de los requisitos. De manera inicial se da soporte para la librería *CAKEIndexer* pero, dada la arquitectura del sistema, es fácilmente sustituible por otro módulo en el caso de que exista y cumpla con un contrato común. Esta librería nos permite obtener fácilmente los recuentos asociados a un determinado texto, en nuestro caso, una descripción de requisito. Para ello se hace uso de la siguiente estructura:

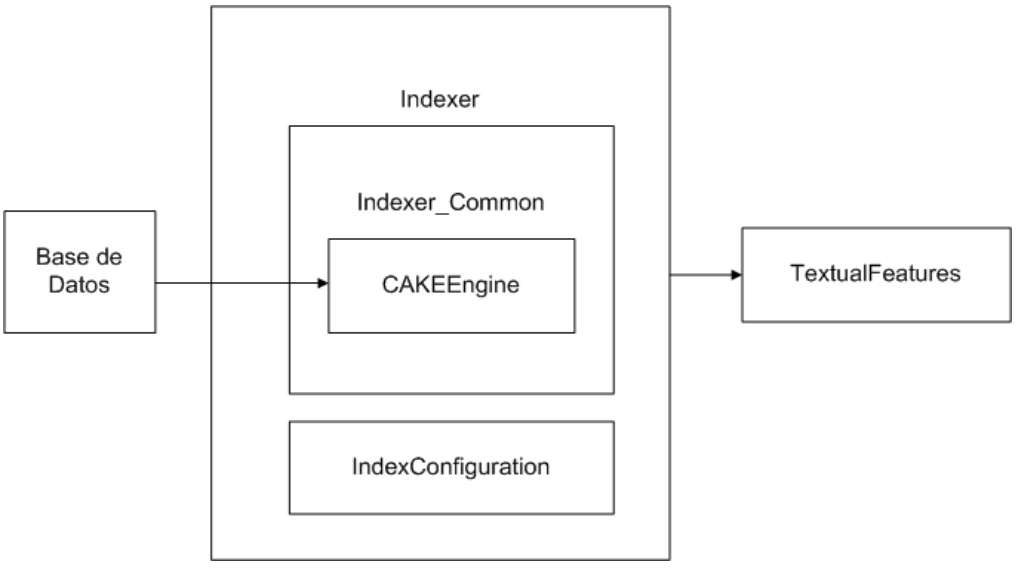

Ilustración 17: Flujo de información desde la Base de Datos hacia las clases de CAKE Indexer y de aquí hacia el sistema desarrollado

Como se ve en la Ilustración 17, existe un elemento fundamental que sería el *Indexer*. Dentro de esta clase se encuentran, entre otros, dos clases importantes. Por un lado, está el *IndexConfiguration* que guarda parámetros de configuración útiles para el indizador como el idioma en el que se encuentra configurado dicho indizador. Por otro lado, tendría un *Indexer_Common*, que es el que albergará una instancia del *CAKEEngine* y que es el que mantiene toda la información relevante para la identificación de elementos que se ha cargado desde la base de datos que actúe de fuente de datos a *CAKE Indexer*. El objeto que se obtiene

una vez se realiza la indización de un texto es un objeto de tipo *TextualFeatures*, que contendrá cada una de las métricas del texto proporcionadas.

Otro de los componentes importantes en CAKE Indexer es su base de datos que, en la versión auditada, se trata de la revisión 1.3.7. Desarrollada como una base de datos Microsoft Access en su versión 97-2003, contiene toda la información referente al vocabulario y reglas sintácticas que es utilizada para el análisis de textos.

Para la utilización de esta librería, será necesario definir un objeto de cada una de las clases anteriormente comentadas y obtener las características del texto a través del método *getTextualFeatures()* de la clase *Indexer,* que proporciona un objeto de tipo *TextualFeatures* con los recuentos obtenidos para un texto dado. Este objeto posteriormente se convertirá a un objeto de la clase *RequirementFeatures* que almacena la misma información.

El último componente, *Exporter*, se encarga de la importación y exportación de objetos de la aplicación o informes a los diferentes formatos soportados, de forma que el formato de las exportaciones sea fácilmente extensible.

5.2 Especificación del diseño de componentes

Se muestra a continuación cada uno de los componentes involucrados en el sistema. Para cada una de las tres capas definidas en el punto anterior se definirán los diferentes subsistemas o paquetes que la forman. Para cada uno de estos componentes se indicará su identificador, propósito, dependencias con otros componentes y subordinados. Por subordinados se entenderán aquellos componentes que forman parte del componente definido. Así, un subsistema estará formado por varios paquetes y un paquete tendrá como subordinadas las diferentes clases que lo forman (que no serán estudio de este documento).

5.2.1 Componente View

Tipo	Componente principal del sistema
Propósito	Este componente es el contenedor de toda la interfaz gráfica del sistema. Los componentes subordinados se encargan de recibir peticiones por parte de los usuarios a través de los formularios de entrada y transmitir esos comandos al interior del sistema. Del mismo modo se encarga de presentar los resultados al usuario.
Subordinado	Los siguientes paquetes agrupan ventanas y diálogos por función o temática: • *View.Metric*: Se encarga de las ventanas que tienen que ver con la gestión de métricas y componentes asociados, así como de la generación de las mismas. • *View.Main*: Se encarga de la visión principal de la aplicación y recursos auxiliares al mismo.

	• *View.Project*: Se encarga de las ventanas para la gestión de requisitos y proyectos, así como sus datos asociados. • *View.Report*: Se encarga de las ventanas de generación de informes y visualización de estadísticas. • *View.User*: Se encarga de la gestión de las ventanas dependientes de los datos asociados al usuario que inicia sesión en el sistema.
Dependencias	Este subsistema tiene dependencias únicamente con la capa de Controlador, con el fin de traspasar las peticiones y obtener las salidas que deben ser mostradas al usuario. Del mismo modo, contiene una dependencia inherente con el paquete *Windows Forms*, dado que se encarga de la construcción de interfaces de escritorio para sistemas operativos *Windows*.

Tabla 10: Especificación del componente View

5.2.1.1 Componente View.Metric

Tipo	Componente ejecutable
Propósito	Este componente agrupa las ventanas y diálogos relacionados bien con el proceso de evaluación de proyectos o bien con el proceso de obtención de métricas de evaluación así como de la gestión de las métricas y elementos asociados a las mismas.
Subordinado	Este componente no dispone de ningún componente subordinado, sino que se trata directamente, de un contenedor de las clases que se ajustan a la descripción anteriormente comentada.
Dependencias	Este componente mantiene la dependencia con la capa de Controlador, con el fin de traspasar las peticiones y obtener las salidas que deben ser mostradas al usuario. Del mismo modo, mantiene la dependencia inherente con el paquete *Windows.Forms*, dado que se encarga de la construcción de interfaces de escritorio *Windows*.

Tabla 11: Especificación del componente View.Metric

5.2.1.2 Componente View.Main

Tipo	Componente ejecutable
Propósito	Este componente agrupa las ventanas y diálogos relacionados con la ventana principal de la aplicación y pequeños elementos auxiliares para completar la funcionalidad de la misma.
Subordinado	Este componente no dispone de ningún componente subordinado, sino que se trata directamente, de un contenedor de las clases que se ajustan a la descripción anteriormente comentada.
Dependencias	Este componente mantiene la dependencia con la capa de Controlador, con el fin de traspasar las peticiones y obtener las salidas que deben ser mostradas al usuario. Del mismo modo, mantiene la dependencia inherente con el paquete *Windows.Forms*, dado que se encarga de la construcción de interfaces de escritorio *Windows*.

Tabla 12: Especificación del componente View.Main

5.2.1.3 Componente View.Project

Tipo	Componente ejecutable
Propósito	Este componente agrupa las ventanas y diálogos relacionados con la con la visualización de datos asociados a un proyecto y sus elementos asociados.
Subordinado	Este componente no dispone de ningún componente subordinado, sino que se trata directamente, de un contenedor de las clases que se ajustan a la descripción anteriormente comentada.
Dependencias	Este componente mantiene la dependencia con la capa de Controlador, con el fin de traspasar las peticiones y obtener las salidas que deben ser mostradas al usuario. Del mismo modo, mantiene la dependencia inherente con el paquete *Windows.Forms*, dado que se encarga de la construcción de interfaces de escritorio *Windows*.

Tabla 13: Especificación del componente View.Project

5.2.1.4 Componente View.Report

Tipo	Componente ejecutable
Propósito	Este componente agrupa las ventanas y diálogos relacionados con la exportación y generación de informes.
Subordinado	Este componente no dispone de ningún componente subordinado, sino que se trata directamente, de un contenedor de las clases que se ajustan a la descripción anteriormente comentada.
Dependencias	Este componente mantiene la dependencia con la capa de Controlador, con el fin de traspasar las peticiones y obtener las salidas que deben ser mostradas al usuario. Del mismo modo, mantiene la dependencia inherente con el paquete *Windows.Forms*, dado que se encarga de la construcción de interfaces de escritorio *Windows*.

Tabla 14: Especificación del componente View.Project

5.2.1.5 Componente View.User

Tipo	Componente principal del sistema
Propósito	Este componente agrupa las ventanas y diálogos relacionados con el usuario que ha iniciado sesión en el sistema y aquellos datos que son propios del mismo.
Subordinado	Este componente no dispone de ningún componente subordinado, sino que se trata directamente, de un contenedor de las clases que se ajustan a la descripción anteriormente comentada.
Dependencias	Este componente mantiene la dependencia con la capa de Controlador, con el fin de traspasar las peticiones y obtener las salidas que deben ser mostradas al usuario. Del mismo modo, mantiene la dependencia inherente con el paquete *Windows.Forms*, dado que se encarga de la construcción de interfaces de escritorio *Windows*.

Tabla 15: Especificación del componente View.User

5.2.2 Componente Controller

Tipo	Componente de primer nivel que conforman el primer nivel de descomposición del sistema
Propósito	Este componente se encarga de recibir las peticiones por parte de los componentes que forman parte de la vista y verificar que los datos recibidos cumplen las exigencias en cuanto a tamaño y contenido exigidos en función de la funcionalidad. La razón para concentrar esta funcionalidad en el controlador en lugar de hacerlo directamente en la vista, es la de reducir dentro de lo posible la lógica y responsabilidades de la vista, de forma que, si en un futuro se buscase realizar una sustitución de la interfaz por otra, bien web mediante ASP.NET o bien mediante WPF, el nivel de dificultad sería mucho menor. Las peticiones que cumplen las exigencias son trasladadas a la capa del modelo con el fin de que sean resultas. Finalmente, es el encargado de transmitir la solución del modelo al componente de la vista que ha emitido la petición.
Subordinado	Este componente no tiene a su vez ningún componente subordinado
Dependencias	Este componente tiene un conjunto de dependencias con el componente del modelo, al cual transmite las peticiones válidas que ha obtenido a través de la vista.

Tabla 16: Especificación del componente Controller

5.2.3 Componente Model

Tipo	Componente de primer nivel que conforman el primer nivel de descomposición del sistema
Propósito	Este componente se encarga de cumplir con la funcionalidad del sistema. Recibe a través del controlador unas peticiones y a través de las clases asociadas se encarga de cumplir con la funcionalidad solicitada en la aplicación.
Subordinado	Este componente tiene a su vez otros dos componentes subordinados que se encarga de cumplir con la funcionalidad asociada a cada uno de los dos campos básicos del sistema, la organización de requisitos y la generación de métricas: • Componente *Model.RequirementData*: Este componente se encarga de contener las clases y cumplir las funcionalidades asociadas a la gestión y organización de requisitos. • Componente *Model.MetricDats*: Este componente se encarga de contener las clases y proporcionar la funcionalidad asociada a la obtención y generación de métricas de evaluación.
Dependencias	Este componente a su vez contiene una dependencia con la capa inferior, la capa de infraestructura con el fin de exportar alguno de los objetos u otorgarles persistencia. También es necesaria la comunicación con el módulo de obtención de recuentos asociados a la descripción de los requisitos.

Tabla 17: Especificación del componente Model

5.2.3.1 Componente Model.MetricData

Tipo	Componente de segundo nivel que se encuentra dentro del componente *Model*
Propósito	Este componente posee el conjunto de clases asociados con la obtención de métricas y gestión de experimentos asociados. Ofrece las funcionalidades para la obtención de nuevas métricas de evaluación. Del mismo modo, contiene también la lógica que ejecuta la aplicación del algoritmo genético, así como el hilo de obtención de recuentos.
Subordinado	Este componente no dispone de ningún componente subordinado, sino que se trata directamente, de un contenedor de las clases que se ajustan a la

	descripción anteriormente comentada.
Dependencias	Este componente mantiene dependencias tanto con la capa inferior, es decir, la capa de infraestructura para comunicarse con los componentes externos y mantiene una dependencia con el otro componente del modelo, el componente *RequirementData* ya que existen relaciones entre ambos componentes.

Tabla 18: Especificación del componente Model.MetricData

5.2.3.2 Componente Model.RequirementData

Tipo	Componente de segundo nivel que se encuentra dentro del componente *Model*
Propósito	Este componente posee el conjunto de clases asociados con la gestión de proyectos y requisitos. Ofrece las funcionalidades para la gestión de requisitos y proyectos asociados.
Subordinado	Este componente ya no dispone de más componentes subordinados, sino que ya se trata de un contenedor de clases. Entre las clases que contiene se encuentran las siguientes:
Dependencias	Este componente mantiene dependencias tanto con la capa inferior, es decir, la capa de infraestructura para comunicarse con los componentes externos y mantiene una dependencia con el otro componente del modelo, el componente *MetricData* ya que existen relaciones entre ambos componentes.

Tabla 19: Especificación del componente Model.RequirementData

5.2.4 Componente Infraestructure

Tipo	Componente de primer nivel que conforman el primer nivel de descomposición del sistema
Propósito	El propósito de este componente es abstraer la lógica de la comunicación con componentes externos del propio modelo de datos de la aplicación. El objetivo es permitir la comunicación con esos componentes externos a través de una “interfaz nueva” que no se vea afecta por el cambio de alguno de esos componentes.
Subordinado	El componente tiene a su vez, tres componentes subordinados, cada uno de los cuales tienen como función específica la comunicación con un componente externo diferente:

	• *Infraestructure.Database*: Este componente se encarga de la comunicación con la base de datos del sistema tanto para obtener como para actualizar datos. • *Infraestructure.Exporter*: Este componente se encarga de la importación y exportación de los diferentes objetos del sistema a los formatos permitidos. Del mismo modo, se encarga de la generación de los informes. • *Infraestructure.CAKE*: Este componente se encarga de la comunicación con el módulo de obtención de los recuentos asociados a la descripción de un requisito, en el caso inicial, *CakeIndexer* para obtener los valores de dichos recuentos.
Dependencias	Este componente tiene dependencias con cada uno de los componentes externos con los cuales se comunica, como son *CakeIndexer* y el sistema gestor de base de datos.

Tabla 20: Especificación del componente Infraestructure

5.2.4.1 Componente Infraestructure.Database

Tipo	Componente de segundo nivel que se encuentra dentro del componente *Infraestructure*
Propósito	Este componente se encarga de la comunicación con la base de datos del sistema tanto para obtener como para actualizar datos.
Subordinado	Este componente no dispone de ningún componente subordinado, sino que se trata directamente, de un contenedor de las clases que se ajustan a la descripción anteriormente comentada.
Dependencias	Este componente mantiene dependencias con el sistema gestor de base de datos a través del cual se quiere dar persistencia al sistema.

Tabla 21: Especificación del componente Infraestructure.DataBase

5.2.4.2 Componente Infraestructure.Exporter

Tipo	Componente de segundo nivel que se encuentra dentro del componente *Infraestructure*
Propósito	Este componente se encarga de la importación y exportación de los diferentes objetos del sistema a los formatos permitidos. Del mismo modo, se encarga de la generación de los informes.
Subordinado	Este componente no dispone de ningún componente subordinado, sino que se trata directamente, de un contenedor de las clases que se ajustan a la descripción anteriormente comentada.
Dependencias	Este componente no mantiene ninguna dependencia con ningún componente relacionado con el sistema

Tabla 22: Especificación del componente Infraestructure.Exporter

5.2.4.3 Componente Infraestructure.CAKE

Tipo	Componente de segundo nivel que se encuentra dentro del componente *Infraestructure*
Propósito	Este componente se encarga de la comunicación con el módulo de obtención de los recuentos asociados a la descripción de un requisito, en el caso inicial, *CakeIndexer* para obtener los valores de dichos recuentos.
Subordinado	Este componente no dispone de ningún componente subordinado, sino que se trata directamente, de un contenedor de las clases que se ajustan a la descripción anteriormente comentada.
Dependencias	Este componente mantiene dependencias con el componente de obtención de recuentos, en nuestro inicial, *CakeIndexer*.

Tabla 23: Especificación del componente Infraestructure.CAKE

6 Detalles de la implementación

En este capítulo se pasarán a desgranar aquellos aspectos que sean más relevantes en la tarea de la implementación de este proyecto. Dado que el número de funcionalidades y la magnitud del proyecto es suficientemente grande, se ha desechado la opción de realizar una descripción formal del diseño detallado y se ha decidido comentar aquellos factores que requieran una explicación adicional o tengan una mayor importancia dentro del funcionamiento del sistema.

6.1 Implementación del algoritmo genético

Uno de los aspectos más relevante dentro de la implementación del código del presente Proyecto Fin de Carrera, es la implementación del algoritmo genético que permite obtener diferente métricas de evaluación de requisitos. Para el correcto entendimiento de la implementación que se ha realizado en este proyecto del algoritmo genético es necesario detallar las diferentes fases que se han hecho referencia en *2.3 Algoritmos genéticos*.

El objetivo del algoritmo genético, tal y como se ha comentado, es la obtención de "buenas" métricas de evaluación y para ello es necesario determinar los parámetros de las métricas mediante la comparación con las medidas de calidad asignadas a los diferentes miembros del conjunto de entrenamiento [Genova 08a]. En el caso particular de este proyecto fin de carrera, las medidas externas se correspondientes a calificaciones académicas que el equipo docente de las asignaturas Ingeniería del Software I e Ingeniería del Software II han adjudicado a los diferentes proyectos que han realizado sus alumnos en los últimos cuatro años.

Cada métrica "candidata" estará compuesta por un conjunto de campos que son los siguientes:

- El tipo de índice de calidad (promedio, mínimo, mixto)
- Para cada indicador: factor de ponderación, predominancia, extremos de los intervalos.

A la hora de detallar la implementación del algoritmo genético es necesario establecer previamente una nomenclatura con el fin de que el lector pueda seguirla de la manera más fácil y cómoda posible, la cual se expone en la Tabla 24:

Nomenclatura
L proyectos P_j
N_j requisitos por proyecto r_{ij}
M recuentos e indicadores x_{ijk}, $q_k(x_{ij})$
Operador para obtener el índice de calidad de un requisito Ω
Factor de ponderación de un indicador w_k
Predominancia del indicador d_k
Extremos de los intervalos e_{1k}, e_{2k}, e_{3k}, e_{4k}

Tabla 24: Explicación de la nomenclatura

6.1.1 Preparación del experimento

La primera etapa del algoritmo genético es la preparación del experimento o lo que es lo mismo, la inicialización del mismo, para ello es necesario que el usuario introduzca una serie de datos que serán utilizados en el ámbito de la ejecución del algoritmo:

Los diferentes campos a los cuales el usuario debe asignar valor, son los siguientes:

- **Corpus de proyectos del conjunto de entrenamiento sobre los que se va a aplicar el experimento**. Es un conjunto de L proyectos P_j, donde cada proyecto P_j consta de una calificación académica CA_j y un número variable de requisitos textuales N_j.
- **Valores nominales de los niveles de calidad.** Para los diferentes niveles nominales de calidad Bueno, Medio y Malo, será necesario asignarle un valor numérico, como por ejemplo 2,1 y 0.
- **Selección de los M indicadores que van a determinar las diferentes métricas**. Es decir, que indicadores de los disponibles se van a utilizar para generar las métricas.
- **Tipo de medida global**. Es necesario cual será la medida global que será utilizada para la comparación con la medida de calidad asignada por el evaluador humano. Los valores posibles son los indicados en 2.1.6.2 Medida global de calidad de un conjunto de requisitos.
- **Tamaño de la población de individuos S**. Establece cual será el tamaño de la población en número de individuos. Su valor por defecto será 200.
- **Número máximo de generaciones G**. Establece el número máximo de iteraciones que se ejecutará el algoritmo en búsqueda de la métrica óptima para el conjunto de entrenamiento especificado. Su valor por defecto será de 10.000 generaciones.
- **Número de generaciones consideradas "recientes" R**. Establece el número de generaciones en el que se debe mantener el fitness de la generación, por debajo del umbral, para la finalización del experimento. Su valor por defecto es 100 generaciones y tendrá como cota superior siempre el número máximo de generaciones ($R \leq G$).
- **Umbral de parada U para el *fitness.*** Establece el umbral de parada para el experimento, siendo su valor por defecto 0.001.
- **Fracción de progenitores reproductores P.** Establece el porcentaje de individuos que serán utilizados para la generación de nuevos individuos por sobrecruzamiento (valor por defecto, 60%).

- **Tamaño del torneo T**. Establece el número de individuos que participarán en cada uno de los torneos de los cuales se seleccionará un ganador, su valor por defecto será 5 y siempre tendrá como cota superior el número de individuos de la población (T≤S).

- **Probabilidad de mutación μ**. Establece la probabilidad en tanto por uno, de que los individuos generados por sobrecruzamiento muten en alguno de sus gente. Su valor por defecto será el inverso del tamaño del individuo medido en genes, (τ).

- **Valores máximo y mínimo de la medida externa de calidad**. Se tratan de valores introducidos por el usuario que establecerán aquellos valores que actuarán como extremos de la medida externa de calidad con el fin de normalizarlos en la misma escala que la medida global proporcionada por las métricas.. (CA_{max} y CA_{min}).

Una vez especificados los valores que identifican y definen a un experimento, la aplicación necesita realizar una serie de cálculos y procedimientos previos al comienzo de la ejecución del algoritmo genético.

- Normalización de las medidas de calidad externas en el rango de los valores nominales de los niveles de calidad: $CA_j \rightarrow A_j$.

  Para la realización de la normalización será uso de la siguiente fórmula:

$$A_j = \frac{(CA_j - CA_{\min}) \times (Nom_{bueno} - Nom_{malo})}{(CA_{max} - CA_{\min})}$$

- Configuración del módulo de cálculo de recuentos para el cálculo especifico, suministrando la información necesaria, especialmente las listas de palabras que deberán ser utilizadas para satisfacer a los indicadores que así lo requieran.

- Obtención de los recuentos numéricos a partir de los requisitos (x_{ijk}). El sistema obtiene los recuentos

- Obtención de los *M* valores máximos obtenidos para cada recuento numérico k en el conjunto de todos los requisitos, denotado MAX_k. Estos valores serán utilizados para las mutaciones.

6.1.2 Función de adaptación (*fitness*) y criterio de parada

El objetivo del experimento es minimizar la distancia entre las medidas de calidad y las calificaciones académicas. Así pues, en cada generación g se define el *fitness* de un individuo como el promedio en los L proyectos P_j del valor absoluto de la diferencia entre la medida global de calidad del proyecto y la medida normalizada A_j. El *fitness* de una generación es el mínimo de todos sus individuos, debido a que el mejor individuo es el que la diferencia con las medidas externas sea la mínima posible:

$$F_g = \underset{t=1}{\overset{S}{MIN}}\left(\frac{1}{L}\sum_{j=1}^{L}\left|Q_j - A_j\right|\right)$$

Ecuación 1: Definición del fitness de una generación

El *fitness* parcial del experimento F_g' es el mínimo del obtenido en las primeras *g* generaciones. El *fitness* del experimento, *F*, es el mínimo del obtenido en todas las generaciones[3].

El experimento se detiene cuando se alcanza el número máximo de generaciones *G*, o antes si se ha alcanzado el umbral de parada *U* en todas y cada una de las *R* últimas generaciones.

6.1.3 Composición del individuo y población inicial

Cada individuo especifica la forma de calcular el índice de calidad de un requisito individual, es decir, establece los parámetros del operador Ω. El individuo tendrá un tamaño variable en función del número de indicadores seleccionados. El primer gen del individuo especifica el tipo de índice de calidad utilizado (promedio, mínimo, mixto).

A continuación vienen los genes que especifican los indicadores, en grupos de tres. El primer gen de un indicador especifica el factor de ponderación, el segundo gen especifica si un indicador es predominante o no (este gen sólo tiene efecto si el operador Ω es mixto), y el tercer gen (dividido a su vez en dos o cuatro subgenes) especifica los intervalos de la función escalonada de transformación. Los indicadores crecientes o decrecientes requieren dos subgenes para especificar los intervalos, mientras que los indicadores convexos o cóncavos requieren cuatro subgenes.

El tamaño del individuo es $\tau = 1 + 3*M$. Así, un experimento con cinco indicadores crecientes y uno convexo tiene un tamaño τ de 19 genes (si se cuentan los subgenes como genes, entonces la cuenta se eleva a 27 genes/subgenes). Para el cálculo de la probabilidad de mutación no se cuentan los subgenes, sino los genes.

Los rangos de los genes son:

- Tipo de índice de calidad Ω: promedio, mínimo, mixto.
- Factor de ponderación normalizado de un indicador w_k: valores enteros 0...100.
- Predominancia del indicador d_k: *booleano*.
  Extremos de los intervalos e_{1k}, e_{2k}, e_{3k}, e_{4k}: valores enteros 0...MAX_k.

Los genes de la población inicial de S individuos se calculan de forma aleatoria, con distribución uniforme en el rango de valores. Los M factores de ponderación w_k se normalizan antes de copiarlos al individuo, para que se cumpla que la suma es 100.Si no se normalizasen los pesos, podríamos obtener individuos equivalentes (por ejemplo, con pesos 20-80-35 y pesos 40-160-70). Los extremos de los intervalos se ordenan de menor a mayor antes de copiarlos al individuo.

---

[3] F_g' = F*g* porque el mejor individuo de una generación siempre pasa a la siguiente, salvo que el porcentaje de selección/reemplazamiento sea el 100%. Así pues, el *fitness* de la generación, y por tanto el *fitness* parcial del experimento, es monótonamente decreciente, salvo que el porcentaje de selección sea el 100%

6.1.4 Selección

La selección de individuos para reproducción se realiza mediante el método de torneos de tamaño *T*. En cada torneo se seleccionan aleatoriamente *T* individuos, y gana el mejor de ellos. Si hay empate, se selecciona aleatoriamente un y sólo un ganador. El ganador puede volver a participar en cualquier torneo posterior.

El proceso se repite *P*S* veces hasta que se alcanza la fracción de individuos reproductores, o progenitores. En principio puede suponerse que cuanto más grande es T más selectivo es el proceso, y por tanto converge más rápidamente; esto no es necesariamente bueno, ya que el peligro de converger hacia mínimos locales es mayor. Análogamente, cuanto mayor es P más individuos peores son reemplazados, y por tanto el proceso converge más rápidamente.

6.1.5 Sobrecruzamiento

Cada par de individuos reproductores engendra dos hijos. Los progenitores se emparejan en el orden en que han sido seleccionados. No hay sesgo hacia el emparejamiento de los mejores, porque los ganadores de un torneo vuelven a participar en sucesivos torneos. Cada uno de los hijos se forma eligiendo aleatoriamente, con igual probabilidad, el gen del padre o el gen de la madre para cada posición. Los genes que especifican los intervalos de la función escalonada de transformación para cada indicador se transmiten íntegramente de padres a hijos (es decir, los dos o cuatro subgenes en bloque).

6.1.6 Mutación

Cada nuevo individuo es sometido a mutación. Cada gen muta con probabilidad μ. La probabilidad de que un individuo no mute viene dada por $(1-\mu)^{\tau}$. Si $\mu=1/\tau$, y τ es suficientemente grande, esto es aproximadamente igual a 1/e (36,6%).

Cuando un gen muta, se escoge aleatoriamente un valor dentro de su rango. Cuando el gen que muta es el que especifica los intervalos, entonces se escoge aleatoriamente uno de los dos (cuatro) subgenes y se le da aleatoriamente un nuevo valor; se reordenan los dos (cuatro) valores resultantes, y así se obtiene una nueva especificación válida de la función escalonada de transformación.

Antiguo	**1-3-5-8**
Mutado	1-2-5-8
Reordenado	1-2-3-8

Tanto el sobrecruzamiento como la mutación pueden afectar a los M factores de ponderación w_k, por tanto al llegar a este punto es necesario renormalizar todos ellos, para que siga cumpliéndose que la suma es 100. Si la renormalización se realiza una vez que han mutado todos los pesos, entonces no hay sesgo a favor de ninguno de ellos.

6.1.7 Reemplazamiento de individuos

De la población original se descarta la fracción *P* de individuos con peor *fitness*, hayan sido padres o no, que son sustituidos por los *P*S* hijos engendrados en esta generación, con lo que la nueva generación vuelve a tener *S* individuos. Así pues existen las siguientes posibilidades: un individuo se reproduce o no, un individuo pasa o no a la siguiente generación; las cuatro combinaciones son posibles.

6.1.8 Presentación de resultados

De cada generación que cumpla *alguna de las siguientes condiciones* se muestra por pantalla:

- Es la primera generación, así podemos ver en qué estado comenzamos el experimento
- Es una generación que mejora el *fitness* parcial alcanzado hasta la generación anterior $F_g < F_{g-1}$'
- Es una generación que "entra" bajo umbral ($F_g <= U$) y ($F_{g-1} > U$)
- Es una generación que "sale" del umbral ($F_{g-1} <= U$) y ($F_g > U$)
- Es la última generación, por alcanzar el criterio de parada o el límite de generaciones del experimento

Los datos que se mostrarán de cada una de esas generaciones son los siguientes:

- Fecha y hora.
- Número de generación, g.
- *Fitness* de la generación, F_g.
- Número de generaciones consecutivas en las que se ha alcanzado el umbral de parada U.
- Estructura del individuo ganador (o individuos, si hay empate entre varios).

A continuación se muestran los resultados que serían presentados en un proceso real.

Fecha y hora	Gen.	Fitness	Bajo umbral	Estructura de los mejores individuos
2008-01-04 12:00:00	1	1,4167	0	individuo 1 (explicitar aquí todos los genes)
2008-01-04 13:05:17	506	1,2290	0	individuo 506
2008-01-04 13:06:21	507	1,1073	0	individuo 507a (tres empatados en esta generación) individuo 507b individuo 507c
2008-01-04 16:10:35	2043	0,3062	0	individuo 2043
2008-01-04 22:06:21	5028	0,0009	1	individuo 5028
2008-01-04 22:07:01	5033	0,0007	6	individuo 5033 (no parada)
2008-01-04 22:08:04	5040	0,0014	0	individuo 5040 (sale de debajo del umbral)
2008-01-05 01:03:12	7451	0,0008	1	individuo 7451
2008-01-05 01:14:29	7550	0,0008	100	individuo 7550 (parada)

Tabla 25: Ejemplo de resultados mostrados para U=0,001 y R=100

Cabe destacar que el ejemplo que se muestra es en un caso de reemplazo total. En el caso de que el reemplazo no fuera total, el *fitness* es monótonamente decreciente, y se alcanzaría el criterio de parada en la generación 5127 con un *fitness* de 0,0007.

6.2 Gestión de permisos y objetos compartidos entre usuarios

Dada la naturaleza colaborativa de la aplicación y la cantidad de diferentes entidades que entran en juego y se han detallado en el modelo conceptual en la sección 4.1 Modelo conceptual, cabe realizar un extenso comentario acerca de cómo se realiza la gestión de permisos entre estos diferentes objetos.

En general, el definidor/administrador de una entidad es el que puede crearla o eliminarla, así como conceder o retirar permisos sobre la misma, mientras que para vincular o desvincular hay que ser definidor/administrador de la entidad principal y usador de la subordinada. A continuación se desgranan los tipos de permisos y las consecuencias de la realización de algunas acciones.

6.2.1 Vincular/Desvincular entidades

El usador de una entidad (métrica o dominio) que es administrador de un proyecto puede vincularla o desvincularla al proyecto; análogamente, el usador de una lista puede crear un recuento asociado a dicha lista y, a su vez asociarla a las métricas cuyo permiso sea de Definidor.

6.2.2 Conceder/Retirar permisos

El administrador de un proyecto puede retirar el permiso a otros escritores o lectores del proyecto, rebajar el permiso de escritor a mero lector, o promocionar un lector a escritor, con los correspondientes efectos en la forma de acceder al proyecto. Sin embargo, no puede modificar el permiso de otros administradores del proyecto.

El definidor de una entidad (dominio/métrica/lista) puede retirar el permiso a un usador que no la esté usando. Del mismo modo, el definidor de una entidad (dominio/métrica/lista) también puede retirar el permiso a un usador que la esté usando. La entidad ya no puede ser vinculada a nuevos proyectos o métricas (deja de ser visible en la lista de selección del usador), pero se mantienen las vinculaciones existentes con entidades principales.

Cualquier usuario con acceso a una entidad puede renunciar a su permiso sobre ese proyecto, siempre que no sea el único administrador (siempre debe quedar al menos un administrador). Análogamente, el definidor de una entidad (dominio/métrica/lista) puede renunciar a su permiso sobre esa entidad, siempre que no sea el único definidor.

El escritor o lector de un proyecto puede renunciar a su permiso sobre ese proyecto. Análogamente, el usador de una entidad puede renunciar a su permiso sobre ella, con los mismos efectos y restricciones que si fuera el definidor quien retirase el permiso.

6.2.3 Crear/Eliminar entidades

El administrador de un proyecto puede eliminarlo por completo junto con todos los permisos concedidos sobre el mismo a otros escritores o lectores; dada la gravedad de la

acción, se solicita confirmación al administrador antes de ejecutar la eliminación, incluso pidiendo que vuelva a introducir su contraseña.

El definidor de una entidad (dominio/métrica/lista) puede eliminarla si no hay ningún usador con permisos sobre la misma. El definidor de una entidad (dominio/métrica/lista) sí puede eliminarla aunque haya usadores con permisos sobre la misma, o aunque la entidad esté vinculada a otra entidad principal. La entidad ya no puede ser vinculada a nuevos proyectos o métricas (deja de ser visible en la lista de selección del usador), pero se mantienen las vinculaciones existentes con entidades principales.

6.2.4 Gestión de permisos en experimentos

Es posible compartir experimentos pero, en este caso, no existen diferentes roles sino que el acceso es total al experimento. Del mismo modo, cuando se comparte un experimento se comparten los proyectos de entrenamiento que se hayan utilizado para la realización de dicho experimento.

6.2.5 Gestión de permisos en los recuentos

Los recuentos no se comparten de la manera estricta que se comparten el resto de elementos comentados, sino que se permite el acceso a recuentos definidos por otro usuario si se encuentran vinculadas a métricas a las cuales se tiene acceso.

6.3 Algoritmo de obtención de recuentos asociado a las descripciones

Otra de las partes menos intuitivas dentro de la aplicación es la utilización de un hilo que se encarga de ir realizando el proceso de obtención de recuentos asociados a las descripciones de los requisitos. Dado que existe un conjunto de diferentes proyectos que son accedidos por varias personas, es necesario, establecer un algoritmo que el realice ese pre-cálculo de recuentos de forma que se minimice el “trabajo sobre-realizado”. Es decir, si dos personas tienen el mismo proyecto, intentar evitar que ambas recalculen los recuentos asociados a las descripciones de los requisitos contenidos en el proyecto.

Para eso se seguirán las diferentes reglas:

- Priorizar los proyectos gestionados por la aplicación frente a los proyectos de entrenamiento
- No recalcular requisitos que tengan valores válidos
- Intentar minimizar el trabajo realizado inútilmente.

Siguiendo esas premisas, el proceso que se sigue es el siguiente:

- Se selecciona aleatoriamente un proyecto gestionado por la aplicación y se precalculan sus requisitos
- Posteriormente, se sigue el orden alfabético, de forma que el único valor aleatorio es el “proyecto de inicio”.

- Una vez finalizado el precálculo de los proyectos gestionados por la aplicación, se comienza a evaluar los proyectos de entrenamiento en grupos de tres. De modo que en cuando se acaben tres seleccionados aleatoriamente, se vuelve a analizar los proyectos gestionados con el fin de saber si es necesario recalcular algunos requisitos.

Para el precálculo de los recuentos de los requisitos es necesario establecer que recuentos necesitamos, con lo cual es necesario utilizar una métrica para obtener los recuentos. En el caso de los proyectos gestionados por la aplicación se utilizará la métrica por defecto asociada al proyecto, es decir, la primera de las métricas asociadas al proyecto. En cambio, dado que los proyectos de entrenamiento tal y como se puede ver en el diagrama del modelo conceptual no tienen métricas asignadas, se utilizarán los indicadores que estén marcados como activos en el momento actual, que serían los que serían utilizados si se comenzase actualmente un proceso de generación de métricas.

Existen circunstancias en las que es necesario recalcular los requisitos y determinar que sus recuentos no son válidos. A continuación se muestran las circunstancias en las que se invalidarán y recalcularán los requisitos:

- **Se cambia el dominio asignado al proyecto.** De este modo, los recuentos correspondientes a "sustantivos del dominio" y "verbos del dominio" no tienen relevancia y sus valores no tienen por qué ser correctos.

- **Se añade un indicador "basado en lista de palabras" a una métrica**. Dado que el módulo de evaluación obliga a recalcular todos los recuentos a la vez. Si nosotros añadimos un indicador basado en lista de palabras, debemos actualizar la "lista de palabras" en el motor de evaluación y el recuento no está disponible para los requisitos.

- **Se modificar una lista de palabras asociado a un indicador asociado a una métrica**. El recuento asociado a ese indicar no tiene porqué ser correcto, tanto si se añade, elimina o modifica algún término.

- **Se modifica el dominio asociado al proyecto**. Similar al caso anterior, los valores no tienen por qué ser correctos.

- **Cambio en la métrica por defecto asociada a un proyecto**. Dado que la métrica por defecto es la que se utiliza para el precálculo es necesario invalidar esos valores y obtener los nuevos.

7 Fase de Pruebas

En cualquier desarrollo de un producto software es fundamental la fase de pruebas. Dicha fase engloba tanto la definición de las pruebas, como la automatización y ejecución de las mismas, así como la verificación de los resultados obtenidos. Las pruebas deben estar orientadas a construir y entregar un producto de calidad, y son, probablemente, el procedimiento de control de calidad más utilizado.

Una prueba podría definirse como el proceso de ejecución de un programa con la intención de verificar el correcto funcionamiento del sistema ante situaciones anómalas o imprevistas con el fin de detectar y posteriormente resolver error en el desarrollo del producto

Este capítulo pretende mostrar las diferentes pruebas que se han realizado en este proyecto con el fin de intentar garantizar al máximo su calidad. En relación a este punto, conviene tener claros los siguientes aspectos:

- La prueba completa no es posible.
- Dada la naturaleza de este producto, objetivo de un proyecto fin de carrera de la Universidad, el nivel de pruebas alcanzado resulta bastante limitado en comparación a lo que debería ser el plan de pruebas de un producto desarrollado en un entorno profesional.
- Del mismo modo, dada la naturaleza distribuida y paralela de la aplicación, en el sentido de que existen diferentes hilos de ejecución de manera simultánea, las tareas tradicionales de prueba y aseguramiento de calidad del código dejan de tener la efectividad esperada ya que existen casos no reproducibles y un número de posibilidades prácticamente ilimitadas y, por descontado, inabarcable.

Como se indica en el segundo punto, las pruebas realizadas sobre el producto, su planificación, diseño, realización y documentación están limitadas por la naturaleza del contexto en el que se han realizado. Así, este capítulo no pretende ser un plan de pruebas completo de la aplicación, sino un simple resumen o especificación a grandes rasgos de las pruebas que se han podido realizar.

Del mismo modo, dada la naturaleza del proceso de generación de este producto enmarcado en un proyecto fin de carrera, la ‘norma’ o consejo de que las pruebas y las revisiones deben ser realizadas por un equipo externo, es inviable en el sentido de que no existen recursos humanos suficientes para tal fin. De esta manera, no se detallarán una a unas

todas las pruebas realizadas, sino más bien los distintos tipos de pruebas y el porqué de su realización.

En el siguiente punto de este anexo se detallan los diferentes tipos de pruebas que suelen llevarse a cabo sobre un producto, así como su aplicación en este caso concreto. A continuación se presentarán una serie de reflexiones a modo de conclusión del capítulo.

7.1 Tipos de pruebas

Puesto que el objetivo de la prueba es descubrir errores para poder garantizar la calidad del producto, no sólo debe probarse el producto terminado, sino que es necesario, también, realizar pruebas durante el proceso de desarrollo del mismo. Así, por ejemplo debe revisarse la documentación generada. Por ello, podemos afirmar que hay diferentes tipos de pruebas.

7.1.1 Pruebas estáticas

Son aquellas pruebas que analizan el objeto sin tener que ejecutarlo. Suelen denominarse "revisiones". Se suelen realizar, principalmente, en las fases de análisis y diseño. Dentro de este apartado se han realizado las siguientes pruebas:

- **Pruebas estáticas sobre el código.** Se han realizado a lo largo de todo el desarrollo del proyecto software inspecciones sobre el código producido. Dicha tarea ha consistido básicamente en listar casos típicos de error y leer el código simulando estos casos para descubrir posibles errores en dicho código. Esta ha sido una práctica habitual durante la etapa de codificación del sistema, que en muchos momentos se ha combinado con las pruebas unitarias, que se explican más adelante.

- **Pruebas estáticas sobre los documentos.** Se han realizado revisiones y verificaciones de los diferentes documentos generados tanto de manera individual por parte del alumno, como supervisiones por parte del tutor del proyecto, con el fin que la calidad de los documentos fuese la máxima posible.

- **Pruebas estáticas sobre el diseño.** Especial hincapié se ha realizado en la parte de la verificación del diseño, donde se ha comprobado que la estructura definida fuese adecuada para soportar los diferentes elementos mostrados en el modelo conceptual y pueda albergar las funcionalidades indicadas.

7.1.2 Pruebas dinámicas

Son aquellas pruebas que requieren la ejecución del objeto que se está probando. Dentro de este tipo de pruebas, sobre este sistema se han realizado las siguientes:

- **Pruebas unitarias.** Han sido las pruebas más realizadas sobre la aplicación y se ha trabajado con este tipo de pruebas durante todo el proceso de codificación de la misma. El objetivo de estas pruebas es probar de forma individual cada componente del sistema que se desarrolla para comprobar su correcto funcionamiento.

En este proyecto para realizar dichas pruebas se han diseñado, a través, de las posibilidades que ofrece la herramienta utilizada para el desarrollo, Visual Studio en su versión de 2010. Para ello se ha desarrollado un *TestProject*, el cual contendrá todos los elementos para la prueba. Con el fin de satisfacer estas pruebas se ha desarrollado métodos de prueba (*TestMethods*) para aquellos métodos que contengan una lógica que necesite ser 'probada', es decir, se han descartado las *Propiedades* de las clases, los *getters* y *setters*, así como aquellos métodos que tengan una complejidad ciclomática igual a 1, entendiendo como complejidad ciclomática el número de caminos independientes dentro de un fragmento de código y determina la cota superior del número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.

En cada uno de estos métodos se han desarrollado diferentes casos de prueba, variando los datos de entrada, con el fin no sólo de verificar la correctitud de los datos de salida, lo que se considera una *prueba de caja negra*, sino con el fin de probar todas las bifurcaciones de las que se disponga dentro del código (*pruebas de caja blanca*) con el fin de que la cobertura de las pruebas sea la máxima posible. Estas pruebas automatizadas abarcan desde la capa del controlador hasta la infraestructura.

- **Pruebas de integración.** Otro tipo de pruebas fundamentales en el desarrollo del sistema, son las pruebas de integración, las cuales buscan probar la correcta integración entre los diferentes componentes del sistema. Que un componente funcione de manera asilada no significa que vaya a funcionar una vez forme parte del conjunto de la aplicación.

  Para la realización de dichas pruebas se ha utilizado el mismo sistema que en los casos anteriores y se ha realizado mediante la realización de métodos de prueba sobre aquellas clase que actúa de conector entre dos capas del sistema, o que necesitan una interacción con capas inferiores de forma de comprobar que la correctitud de los resultados se mantiene a través de que la información vaya pasando por capas adyacentes y sus resultados siguen siendo igualmente correctos.

- **Pruebas del sistema.** Una vez el sistema está desarrollado pasa a probarse "como un todo". El objetivo es verificar el correcto funcionamiento del mismo una vez están todos sus módulos acoplados y verificar que realiza correctamente todas las funcionalidades especificadas por el usuario. Estas pruebas no tienen una posibilidad de automatización similar a las anteriores, pero de todos modos se ha producido a la utilización de otro de los elementos de prueba conocido como *CodedUITest*, que permite 'grabar' un conjunto de acciones y realizar un conjunto de comprobaciones de alto nivel que se realizan sobre la interfaz con el fin de poder ser reproducidos de manera rápida.

- **Pruebas de implantación.** Una vez se ha probado el correcto funcionamiento del sistema, es necesario comprobarlo en diferentes entornos dado que no es una aplicación de una única instalación, sino que es factible que se ejecute en diferentes ordenadores, con diferentes versiones de software y diferente configuración. Para ello se ha requerido el uso de diferentes ordenadores, donde se ha procedido a la

instalación del sistema y su posterior realización de las mismas pruebas que las que componen las pruebas del sistema con el fin de que la diferente configuración del sistema no influye en la correcta realización de las funcionalidades y objetivos del sistema

- **Pruebas de aceptación.** El objetivo de este tipo de pruebas es que el usuario final de la aplicación pruebe el sistema final y dé el visto bueno al mismo. Dada la particularidad de este proyecto, esta situación ha intentado simularse de manera que el tutor de este proyecto hiciera el papel de cliente, e instalara y probara la aplicación, dando finalmente su visto bueno.

- **Pruebas de regresión**: Estas pruebas están asociadas a la fase de mantenimiento del sistema. El objetivo es comprobar que los cambios efectuados sobre un componente del sistema no introducen un comportamiento no deseado o errores en los componentes no modificados. Estas pruebas forman parte de la fase de mantenimiento, aunque en la realidad este tipo de pruebas se han llevado a cabo durante toda la fase de construcción del sistema.

  Dado a las tareas que se han realizado para la automatización del proceso de pruebas ha permitido que realizado cualquier cambio sobre el sistema se pudiese comprobar la consistencia del producto.

7.2 Ejecución de las pruebas

Dado que la ejecución de las pruebas se realizan mediante un entorno automatizado, con componentes que se encargan de su ejecución y su verificación es necesario disponer de un entorno controlado, en cuanto a la configuración de los datos del sistema en el momento de su prueba.

Para ello, es necesario disponer de un conjunto de datos en la base de datos en el sistema que se cargará para hacer las pruebas. Al igual que el resto de datos, en los datos del proyecto de prueba se proveerá un script SQL con el fin de poder establecer la base de datos a dicho estado.

Del mismo modo, será necesario utilizar un conjunto de ficheros que permitirá realizar tareas de importación así como inicializar datos con el fin de realizar pruebas sobre el conjunto de datos del sistema. Dichos elementos serán los siguientes y los formatos de importación se definen en *Anexo II. Formatos de importación y exportación:*

- **WorkingProject.xml.** Fichero XML que contiene los datos asociados a un *WorkingProject* del sistema.

- **Metric.xml.** Fichero XML que contiene los datos asociados a una métrica.

- **Experiment.xml.** Fichero XML que contiene los datos asociados a un experimento que ha sido previamente exportado.

- **Domain.xml.** Fichero XML que contiene los datos asociaos a un dominio que ha sido previamente exportado.

- **SpecialList.xml.** Fichero XML que contiene los datos asociaos a una lista de términos especiales que había sido previamente exportada.
- **TrainingProjectList.xml.** Fichero XML que contiene los datos asociados a una lista de *TrainingProjects* previamente exportados.
- **Imported.docx.** Fichero Word en su formato de 2007 y que contiene un proyecto en formato textual pero previamente etiquetado.
- **Imported.xls.** Fichero Excel en su formato de 2007 y que contiene un proyecto importable
- **baseReq.mdb**. Fichero Access que contiene un listado de proyectos
- **imported.txt.** Fichero de texto plano que contiene un proyecto importable por la aplicación

Del mismo modo para facilitar la ejecución de pruebas se han creado diferentes listados que contienen los métodos de prueba asociados a cada una de las capas del sistema. Los datos de las pruebas son los siguientes:

Capa	Controlador	Modelo	Infraestructura	Total
Métodos de prueba	123	355	170	648

Tabla 26: Número de casos de prueba por capa

Cabe destacar que cada uno de los métodos de prueba contiene un número de casos de prueba que varía en función de la complejidad ciclomática del método en cuestión o el número de parámetros de entrada de dicho método.

7.3 Conclusión

Como ya se ha comentado en el inicio del capítulo, este documento refleja las pruebas que se han desarrollado sobre el producto, pero no las detalla. Además las pruebas realizadas no han profundizado todo lo que deberían en caso de tratarse de una aplicación comercial dado el tamaño de la aplicación y la condición de proyecto de fin carrera de este sistema.

Así, un verdadero plan de pruebas sobre un sistema de este tipo debería recoger, entre otras cosas, un plan de pruebas, la definición de los casos de prueba y su documentación detallada, la definición de las pruebas realizadas, así como sus resultados documentados, las soluciones a los errores encontrados, etc. Todos estos aspectos no ha sido posible abarcarlos en este proyecto.

A pesar de estas indicaciones se considera que el procedimiento de prueba es lo suficientemente correcto para un proyecto de esta condición y se ha tratado aspectos como la automatización de las mismas y la inspiración formal mediante la realización de pruebas de caja blanca y caja negra, buscando la cobertura de sentencia, con el fin de que las pruebas tengan una gran utilidad.

8 Datos asociados al proyecto

8.1 Planificación

Este proyecto se ha desarrollado durante aproximadamente ocho meses, teniendo su comienzo en Marzo de 2010. Cabe destacar que como actividad anterior del proyecto en sí, se ha realizado un Trabajo Dirigido desde junio 2009 hasta Marzo 2010, consistente en la realización de un análisis y extensión de la librería CAKEIndexer, que ha sido un preámbulo necesario para la realización de este proyecto [Vazquez 10]. A pesar de tratarse de una actividad necesaria para la realización de este proyecto, no se incluye como parte del Proyecto Fin de Carrera, al constituir una entidad propia como Trabajo Dirigido.

Cabe destacar, la dedicación que se ha podido realizar durante los meses con carga lectiva ha hecho que durante esos tres meses hasta Julio el avance en el mismo sea poco y se haya limitado al estudio de documentación, y formulación a requisitos. Es, a partir de la finalización del periodo lectivo, donde se ha podido avanzar de manera drástica en el proyecto entrando propiamente en la fase de implementación. El número de horas dedicadas ha ido variando en función de la disponibilidad del alumno. La distribución de tareas se recoge en el siguiente diagrama de Gantt:

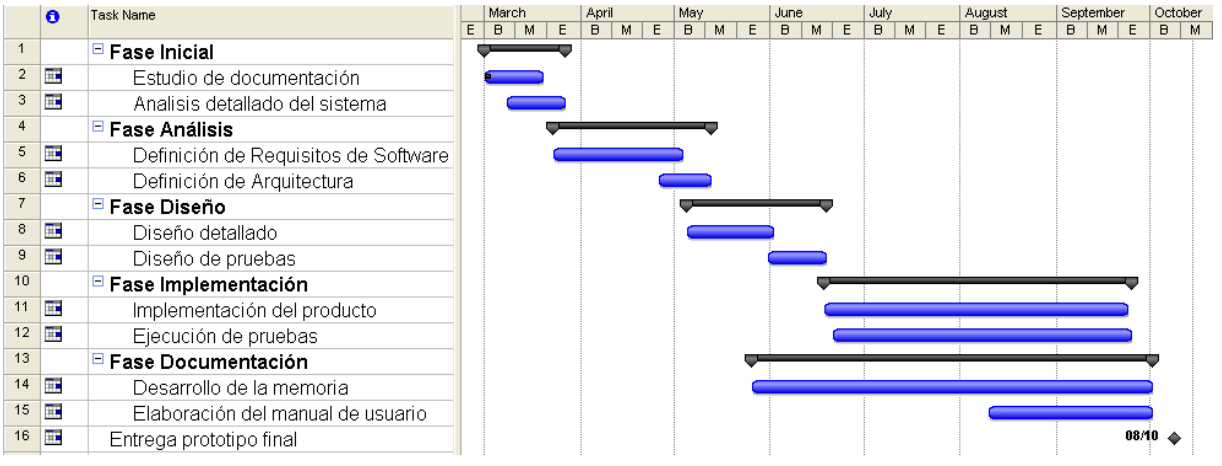

Ilustración 18: Diagrama de Gantt de la realización del proyecto

En una primera etapa se produce el estudio de la documentación proporcionada por parte del tutor, así como documentación relacionada que se incluye en las referencias del presente documento. Posteriormente, a través de diferentes reuniones con el tutor se establece el alcance del proyecto, los límites del mismo y el objetivo a conseguir. Una vez, se han tenido claras las ideas se ha realizado una definición de requisitos (sólo se ha utilizado un

nivel, más detallado, dado a la no existencia de un cliente final con bajo conocimiento técnico), así como una descomposición arquitectónica. Dichos elementos, han sufrido diferentes refinamientos y ajustes en diversas etapas, con lo que no es un modelo "puramente en cascada". Posteriormente, se realiza un diseño más detallado tanto del producto como de las pruebas asociadas al mismo, finalmente se produce la etapa más larga del proyecto, que se trata de la implementación del mismo así como la ejecución de pruebas, lo que repercute en cambios en el código y, ocasionalmente, en el diseño nuevamente sin producir cambios significativos.

Durante todo el proceso de desarrollo del proyecto, se realiza una labor de documentación para la generación del presente documento así como la documentación asociada al producto como es el caso del manual de usuario.

8.2 Herramientas utilizadas

Herramienta	Versión	Descripción
AltovaUModel	**2010**	AltovaUModel es una herramienta de modelado de software que se incluye dentro de la suite desarrollada por la empresa Altova, empresa creadora entre otros productos de XMLSpy. UModel soporta los catorce diagramas especificados en el estándar UML 2. También soporta las funciones de generación de código tanto directa como inversa desde los siguientes lenguajes: Java, C#, Visual Basic, así como la importación y exportación mediante el estándar XMI.
Microsoft Office	**2010**	Microsoft Office 2010 es una última versión de la suite ofimática de Microsoft. Sucesora de Microsoft Office 2007 que incluye compatibilidad extendida para diversos formatos de archivos, así como actualizaciones en la experiencia de usuario.
Microsoft SQL Management Express	**2008**	Microsoft SQL Management Express es una herramienta incluida en la instalación Microsoft SQL Server 2008 con el fin de configurar dicho servidor de base de datos. Incorpora opciones básicas de mantenimiento y gestión, tanto de manera gráfica como mediante comandos.
Microsoft Visual Studio	**2010**	Microsoft Visual Studio es un entorno integrado de desarrollo para sistemas operativo Windows. Soporte diferentes lenguajes de programación como C#, J#, C++, VB.NET o ASP.NET. Visual Studio permite a los desarrolladores crear

		aplicaciones, sitios y aplicaciones web, así como servicios web.
RequirementsStudio	**2.1**	Se trata una aplicación para la gestión de requisitos desarrollada en el seno del grupo de investigación KnowledgeReuse Group, de la Universidad Carlos III de Madrid, en colaboración con la empresa The Reuse Company.

Tabla 27: Listado de aplicaciones utilizadas

8.3 Presupuesto

A continuación, en esta sección se pasará a la realización de un presupuesto que estimará el coste derivado del proyecto en caso de haber sido producido en un entorno profesional. Los diferentes datos de costes son extraídos de la plantilla de presupuestos puesta a disposición de los alumnos de la Universidad Carlos III de Madrid [UC3M]. En ningún caso se pretende establecer el precio que debería pagar un cliente por el desarrollo de este producto ya que no se van a considerar márgenes de riesgo o de beneficio.

8.3.1 Personal

Respecto al personal sólo existe un único desarrollador que ha realizado diferentes roles en el proyecto, desde analista durante la fase de especificación de requisitos, diseñador en la fase de diseño o programador en la fase de implementación. Dado que ha existido una etapa de dedicación parcial desde Marzo 2010 hasta Junio de 2010, se estiman la mitad de hombres mes durante ese tiempo (2 hombres/meses) y se añaden los otros 4 hombres/mes a tiempo completo, dando un total de 6 hombres/mes. El coste hombre/mes es extraído de la plantilla de presupuesto puesto a disposición de los alumnos por la Universidad Carlos III de Madrid. [UC3M].

Apellidos y nombre	Categoría	Dedicación (hombres/mes)	Coste (hombre mes)	Coste Total
Vázquez Vázquez, Alexandre	Ingeniero Informático	6	2.694,39 €	16.166,34 €

Tabla 28: Tabla de coste de personal

8.3.2 Hardware y Software

En la siguiente tabla se detalla el uso de software y hardware necesario para la realización e implantación del proyecto. Nuevamente los valores de "periodo de amortización" y similares son extraídos de la plantilla de presupuesto de la Universidad Carlos III de Madrid [UC3M]. Para el cálculo del coste imputable de cada uno de los elementos hardware y software se hace uso de la siguiente fórmula:

$$\frac{A}{B} \times C \times D$$

A = nº de meses de uso del material.
B = periodo de amortización (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto

Descripción	Coste	%Uso	Dedicación (meses)	Periodo de amortización (meses)	Coste imputable
Ordenador de Desarrollo	600,00 €	100%	8	60	80,00 €
Microsoft Windows XP	200,00 €	100%	8	60	26,67 €
Microsoft Office 2010	285,00 €	100%	8	60	38,00 €
Microsoft Visual Studio 2010	679,00 €	100%	8	60	90,53 €
Microsoft SQL Express 2008	0,00 €	100%	8	60	0,00 €
Altova UModel 2010	300,00 €	100%	8	60	40,00 €
Requirements Studio 2.1	0,00 €	100%	8	60	0,00 €
MySQL 5.03	0,00 €	100%	8	60	0,00 €
				Total	275,20 €

Tabla 29: Coste de material hardware y software

8.3.3 Otros costes

No se han incluido otros costes como dietas o transporte ya que no han sido necesarios en el desarrollo real del proyecto.

8.3.4 Resumen de Costes

Concepto	Coste sin I.V.A
Personal	16.166,34 €
Equipos	275,20 €
Total	16.441,54€

Tabla 30: Resumen de costes sin IVA

El coste total del proyecto antes del aplicar el IVA al coste total asciende a **DIECISEISMIL CUATROCIENTOS CUARENTA Y UN EUROS CON CINCUENTA Y CUATRO CÉNTIMOS**

Presupuesto Costes Totales	
Personal	16.166,34 €
Equipos	275,20 €
Costes Indirectos (IVA 18%)	2.959,48 €
Total	19.401,02 €

Tabla 31: Tabla de resumen de costes

El coste total del proyecto fin de carrera una vez aplicado el IVA ascendería a **DIECINUEVEMIL CUATROCIENTOS UN EUROS CON DOS CÉNTIMOS.**

9 Conclusiones y trabajos futuros

9.1 Situación actual

Una vez concluido el desarrollo de este Proyecto Fin de Carrera, se puede decir que los resultados son satisfactorios ya que se han logrado los objetivos que se buscaban en la realización del mismo.

Por un lado se ha logrado una herramienta que permite combinar las ideas de dos herramientas desarrolladas en el seno de la universidad, como eran Requirements Studio y MeCaReq, de forma que la utilidad de cada una se ve claramente aumentada por el aporte de la otra. Además, se realizan un considerable conjunto de mejoras a las 'partes individuales' de las especificaciones de estas herramientas, con lo que la utilidad del producto es muy alta.

La herramienta cumple con todos los requisitos que se habían formulado en el catálogo de requisitos y se ha intentado proporcionar la mayor fiabilidad mediante la realización de las pruebas, intentando que fuesen exhaustivas y utilizando las librerías proporcionadas por la herramienta de desarrollo con el fin de automatizar las mismas, incluso las que se realizan desde la interfaz, de modo que la utilidad y la capacidad de reproducción de las mismas son muy altas.

Desde el punto de vista del alumno realizador de este proyecto, se considera satisfecho por haber podido trabajar en un proyecto de una gran magnitud (según la herramienta de UNIX para el cálculo de líneas de código, estamos hablando de un proyecto con más de 55.000 de código), así como poder aprender tecnologías nuevas como son las tecnologías pertenecientes al *framework* .NET de la empresa estadounidense Microsoft.

9.2 Dificultades encontradas

A lo largo de la realización de este Proyecto Fin de Carrera se ha tenido que lidiar y superar un conjunto de situaciones adversas que han acontecido durante la realización del proyecto y han influido en la realización del mismo.

Cabe destacar una de las limitaciones impuestas, por parte del tutor, respecto a la tecnología que debía ser utilizada en la implementación del proyecto (.NET y C#).El autor del proyecto no tenía ninguna experiencia en el momento del comienzo del proyecto y sus únicas aproximaciones han sido dos prácticas de dos asignaturas optativas realizada durante la realización de este proyecto. Esto ha supuesto una cierta dificultad inicial que ha debido ser

subsanada mediante documentación y una etapa inicial de “codificación” lenta dada al poco conocimiento del lenguaje.

Del mismo modo, dado que este proyecto debía integrar los resultados de dos proyectos anteriores, *Requirements Studio* [Alonso 08] y *MeCaReq* [Fernandez 08], se ha requerido un estudio de ambas aplicaciones, con el fin de analizar sus puntos fuertes y débiles, aunque muchas de las “deficiencias” ya habían sido reportadas por diferentes usuarios de ambas herramientas, especialmente por el tutor del proyecto.

No obstante, no se ha reutilizado ninguna parte del código de estos dos proyectos con el fin de no caer en los mismos problemas conceptuales y errores que querían ser resueltos por la nueva aproximación del sistema.

Otro de los problemas que se han detectado son ciertas limitaciones existentes en el módulo utilizado para la obtención de los recuentos utilizados para la evaluación de requisitos. Cake Indexer, la librería desarrollada por CISET, aborda un problema muy complejo, la obtención de verbos de un determinado texto. Sin embargo, tal y como se mostró en un trabajo dirigido previo a la realización de este proyecto, sus resultados no son todo lo satisfactorios que podrían ser [Vazquez 10].

9.3 Trabajos futuros

A pesar de que el proyecto aborda las características principales que debían ser resueltas por parte del alumno, existen algunas cuestiones interesantes de cara a una ampliación o continuación del mismo. Entre ellas, destacaremos las siguientes:

- **Evaluación individual de los requisitos y no por proyectos completos**. En el desarrollo actual, el conjunto de entrenamiento del cual disponemos posee una medida de calidad general a un proyecto. Dicha calificación además tiene en cuentas más cosas que estrictamente los requisitos. Una buena forma de mejorar la generación automática de medidas de calidad sería atribuir un valor de calidad a cada uno de los requisitos y no al proyecto completo. El valor de calidad del proyecto podría calcularse como la media de las medidas de calidad de sus diferentes requisitos.

- **Interfaz Web/Móvil:** En el mundo tecnológico en el que vivimos, con conexiones a la red desde gran variedad de dispositivos (móviles, tablets, mp3), anclar una aplicación a un ordenador como único dispositivo es limitar mucho su uso. Dado que los datos se encuentran accesibles desde cualquier lugar, la opción de una interfaz móvil para la aplicación aumentaría mucho su capacidad de uso. Del mismo modo una interfaz web, aprovechando las capacidades del nuevo estándar HTML5 [W3C] para proporcionar una experiencia de usuario próxima a las aplicaciones de escritorio pero con las ventajas de las aplicaciones web es otro de los pasos a considerar seriamente.

- **Migración de CAKE Indexer**. La creación o acoplamiento de otra librería que permita realizar los mismos cálculos que permite *CakeIndexer*, pero de un modo más eficiente e intuitivo sería de gran ayuda y utilidad.

- **Introducción de lógica difusa.** Otro de los aspectos interesantes en el proceso de evaluación es la introducción de lógica difusa en el proceso de generación de métricas. Las funciones escalonadas de transformación utilizadas en el proyecto tienen transiciones abruptas entre los valores Malo-Medio-Bueno. El uso de transiciones difusas es más próximo a la percepción humana y por tanto potencialmente más útil.

10 Referencias

[Alexander 02] Ian Alexander. *Writing Better Requirements*. Addison-Wesley, 2002.

[Alonso 08] Jorge Alonso. Proyecto Fin de Carrera: Requirements Studio. Una herramienta para la gestión de requisitos. Leganés. Universidad Carlos III de Madrid. 2008

[Braude 01] Eric Braude. Software Engineering, *An Object Oriented Perspective*. John Wiley & Sons, 2001.

[CISET] Centro de Innovación y Soluciones Empresariales y Tecnológicas. Recuperado el 21 de Julio de 2010, de http://www.ciset.es

[Domínguez 09] Jorge Domínguez. *The Curious Case of the CHAOS Report 2009*. Recuperado el 21 de Julio de 2010, de http://www.projectsmart.co.uk/the-curious-case-of-the-chaos-report-2009.html

[Fabrini 01] F. Fabbrini, M. Fusani, S. Gnesi, G. Lami. "The Linguistic Approach to the Natural Language Requirements Quality: Benefit of the Use of an Automatic Tool". Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop, pp. 97-105, 2001.

[Fernandez 08] Daniel Fernández. Proyecto Fin de Carrera. Métricas de calidad en requisitos. Universidad Carlos III de Madrid, Leganés. 2008.

[Firesmith 03] Modern Requirements Specification. *Journal of Object Technology*, 1(2) pp 53-64. 2003

[Genova 08a] Gonzalo Génova. Informe Técnico. *Afinamiento en las métricas de calidad de requisitos.*2008

[Genova 08b] Gónzalo Génova. Material docente. Ingeniería del Software I. Universidad Carlos III de Madrid. 2008.

[Genova 08c] G. Génova, J. Fuentes, J. Llorens, O. Hurtado, V. Moreno. Informe Técnico. Métricas de calidad en requisitos. Aprender a escribir buenos requisitos con ayuda de una herramienta consejera. 2008

[Holland 75] Holland, J.H. “Adaptation in natural and artificial systems”. Ann Arbor MI: University of Michigan Press. 1975

[IEEE 830] IEEE Std 830-1998 (Revision of IEEE Std 830-1993). *IEEE Recommended Practice for Software Requirements Specifications* (ieeexplore.ieee.org/iel4/5841/15571/00720574.pdf).

[Kasser 04] Joseph E. Kasser. “The First Requirements Elucidator Demonstration (FRED) Tool”. Systems Engineering 7(3):243-256, 2004.

[Kasser 06] Joseph E. Kasser, W. Scott, X.L. Tran, S. Nesterov. “A Proposed Research Programme for Determining a Metric for a Good Requirement”. *The Conference on Systems Engineering Research*, Los Angeles, California, USA, 2006.

[Pressman 05] Roger S. Pressman. *Software Engineering A Practitioner’s Approach*. McGraw-Hill Pub Co., Sixth Edition 2005.

[Rosenberg 01] Linda H. Rosenberg. “Generating High Quality Requirements”. Proceedings of the AIAA Space 2001 Conference and Exposition, AIAA Paper 2001-4524. American Institute of Aeronautics and Astronautics, Albuquerque, NM, August 28-30, 2001.

[Skinner 10] Skinner, M. *Genetic Algorithms Overview*. Recuperado el 17 de Julio de 2010, de http://geneticalgorithms.aidepot.com/Tutorial/Overview.html

[Sommerville 04] Ian Sommerville. *Software Engineering*,,7th ed. Pearson-Addison Wesley, 2004.

[Tolmos 03] Tolmos Rodríguez-Piñeiro, P. Introducción a los algoritmos genéticos y sus aplicaciones. Universidad Rey Juan Carlos, Madrid, 2003.

[TRC] Recuperado el 21 de Julio de 2010, de http://www.thereusecompany.com

[UC3M] Universidad Carlos III de Madrid. Plantilla de presupuesto del proyecto fin de carrera. 2010. Recuperado el 02 de Octubre de 2010, de http://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera/Formulario_PresupuestoPFC-TFG%20(3)_1.xlsx

[Vazquez 10] Trabajo Dirigido: Evaluación y Extensión de CAKE Indexer. Leganés: Universidad Carlos III de Madrid.

[W3C] HTML 5 Specification Overview. Recuperado el 21 de Julio de 2010, de http://dev.w3.org/html5/spec/Overview.html

[Wilson 97] William M. Wilson, Linda H. Rosenberg, Lawrence E. Hyatt. “Automated Analysis o Requirement Specifications”. *Proceedings of the 19th International Conference on Software Engineering-ICSE’97*,

May 17-23, 1997, Boston, Massachusetts, USA: 161-171.

Anexo I. Glosario

Término	Definición
Algoritmo genético	Se trata de algoritmos pertenecientes a la rama de la inteligencia artificial que se inspiran en la evolución biológica y su base genético-molecular, permitiendo evolucionar una población de individuos sometiéndola a acciones aleatorias similar a lo que ocurre en la evolución biológica.
CAKE Indexer	Se trata de una librería desarrollada por CISET que permite obtener ciertos datos objetivos de un texto mediante un análisis morfológico, sintáctico y semántico que facilita la tarea de la obtención de valores complejos como los verbos de un texto o el número de sustantivos.
Complejidad Ciclomática	Define el número de caminos independientes dentro de un fragmento de código y determina la cota superior del número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez
Dominio	Se trata de un conjunto de términos, generalmente sustantivos, relacionados con un determinado ámbito.
Experimento	Se trata de la configuración con la cual se ha ejecutado el algoritmo genético, tanto los parámetros de configuración del algoritmo, como el conjunto de entrenamientos utilizado, los recuentos a utilizar, y la población inicial del mismo.
Fitness	Elemento asociado a una métrica generada en un proceso de obtención de métricas, permite comprobar su adaptación respecto al conjunto de entrenamiento utilizado.
Indicador	Se trata de cada uno de los componentes de la métrica y es la asociación de un recuento a una métrica con unos parámetros configurables asociados.
Listado de términos especiales	Se trata de un conjunto de términos especiales que permiten detectar ciertas cualidades en un texto como ambigüedad, especulación, incompletitud, etc.
Métrica	Se trata de un elemento utilizado para la evaluación de requisitos. Compuesto por un conjunto de cualidades cuantificables, así como de ciertos intervalos que determinan la calidad para cada uno de ellos permite obtener un valor numérico aproximado a la calidad del requisito.
Paquete	Se trata de un contenedor de requisitos, permitiendo realizar agrupaciones lógicas entre los mismos.

Proyecto	Se trata de un conjunto de requisitos mediante una estructura de paquetes
Proyecto de entrenamiento	Se trata de proyectos que componen el conjunto de entrenamiento y son utilizados en las tareas de la obtención y validación de métricas a través del algoritmo genético
Proyecto gestionado por la aplicación	Se trata de proyectos que son utilizados como parte principal del sistema.
Recuento	Se trata de cada una de las características cuantificables que pueden ser evaluadas dentro de un requisito y que son la base de los indicadores que componen las métricas.
ReqStudio	Se trata de una aplicación desarrollada por Jorge Alonso como proyecto fin de Carrera y que consiste en un organizador de requisitos
TrainingProject	Véase *Proyecto de entrenamiento*
WorkingProject	Véase *Proyecto gestionado por la aplicación*

Anexo II. Formatos de importación y exportación

En este anexo se detallan los formatos de importación y exportación de proyectos con el fin de garantizar la interoperabilidad de *ReqStudio Plus* con otras aplicaciones. Del mismo modo, se facilita diferentes formatos de importación con el fin de que el traspaso de proyectos desde otras aplicaciones sea lo más rápido y cómodo posible para los usuarios.

II.1 Importación/Exportación de proyectos

II.1.1 Modos de importación de proyectos

La aplicación permite un único formato de exportación, pero en cambio permite varios formatos de importación, con dos posibilidades según los casos:

- **Importación de proyectos completo.** En el caso de decantarse por esta opción se importará un proyecto completo, en el cual se importará no sólo el catálogo de requisitos y la estructura de paquetes, sino que se importarán los datos básicos del proyecto (a excepción de los datos automáticos que se sobrescribirán).

- **Importación de requisitos.** En el caso de decantarse por esta opción se permitirá la importación únicamente del catálogo de requisitos, así como la estructura de paquetes y tipos de requisitos asociados, dentro de un proyecto ya preexistente.

Durante la importación de los proyectos gestionados los identificadores de los requisitos serán reenumerados, con lo que su identificador variará para adaptarse al nuevo proyecto que los acoge.

II.1.2 Formato propio

Cabe destacar que *ReqStudio Plus* incorpora su propio formato de definición de un proyecto a través de un fichero XML que cumple con un determinado XML Schema. La exportación e importación de los proyectos propios se realiza a través del framework XMLSeralization propio del framework .NET. Debido a que el modelo de datos diferente de un proyecto gestionado por la aplicación respecto a un proyecto de entrenamiento, es necesario definir diferentes XMLSchema en caso de que se trate de un *TrainingProject*

(proyecto de conjunto de entrenamiento) o un *WorkingProject* (proyecto gestionado por la aplicación).

II.1.3 Otros formatos de importación

En este apartado, se detallarán aquellos formatos a partir de los cuales se pueden añadir proyectos al sistema.

II.1.3.1 Formato propio de *ReqStudio 2.1*

Una de las formas principales para la importación de un proyecto a la aplicación ReqStudio Plus es a través de un fichero *de Microsoft Access* en su versión de 97 a 2003 (*.mdb) que es exportado por la aplicación *RequirementsStudio* en su versión 2.1 fechada en Marzo 2010. Esta forma sólo permite importar proyectos que serán gestionados por la aplicación.

Cabe destacar los datos que serán importados por parte de los contenidos en el documento de importación y cuáles serán sustituidos por valores propios de la aplicación. Se importarán todo el catálogo de requisitos, así como las relaciones existentes entre los requisitos, así como la estructura de paquetes del sistema. Del mismo modo también se importarán los diferentes tipos de requisitos. Respecto a los datos automáticos de un requisito (identificador, versión, creador, último modificador, fecha de creación y fecha de última modificación) no se importarán sino que serán reemplazados, figurando el usuario que realiza la importación como creador y último modificador de cada uno de los requisitos importados, y las fechas de creación y modificación será el momento de la importación. La versión tendrá el valor 1, como versión inicial y no se almacenarán ninguna de las versiones previas existentes, mientras que el identificador se asignará de manera secuencial según se van importando los requisitos.

De los datos propios del proyecto, se respetarán todos a excepción del creador y la fecha de creación los cuales pasarán a ser nuevamente el usuario que realiza la importación y el momento de la importación, respectivamente.

II.1.3.2 Formato Microsoft Excel 2003 y 2007

Otra de las formas de importar un proyecto a la aplicación ReqStudio Plus es a través de un hoja de cálculo realizada a través del software *Microsoft Excel* dentro del paquete *Microsoft Office* tanto en su formato de las versiones 97-2003 (xls) y su versión posterior en la versión de 2007 y siguientes (xlsx). A través de este tipo de documento pueden definirse un número arbitrario de proyectos por documento, es decir, no es necesario un único documento por proyecto. Se permitirá de este modo la importación tanto de proyectos gestionados por la aplicación como proyectos de entrenamiento.

Es necesario establecer un conjunto de ‘normas’ para la importación de proyectos se realice de forma correcta:

- Existirán dos hojas en el fichero de Excel. En la primera de las hojas será en la que se especificarán los datos de cada uno de los proyectos, mientras que en la segunda hoja será donde se detallen los datos referentes a los requisitos.

- Las primeras filas de cada hoja no son leídas y se permite que sean utilizada como cabeceras para cada una de las correspondientes columnas.

- En la primera hoja, aquella donde se definen los proyectos se podrán especificar los siguientes datos que ocuparán las cuatro primeras columnas de la hoja correspondiente:

    - **Identificador:** Se trata de un identificador numérico que sirve para identificar unívocamente al proyecto entre los definidos en el documento en cuestión.

    - **Grupo.** Se corresponde con el grupo al cual pertenece el proyecto. Este campo es útil para la importación de proyectos de entrenamiento y será ignorado en el caso de 'proyectos gestionados' por la aplicación. Será la primera columna de la hoja de cálculo. Se trata de un campo opcional.

    - **Nombre.** Se corresponde con el nombre del proyecto que tendrá el nuevo proyecto que será importado en el sistema. Será la segunda columna de la hoja de cálculo, 'y deberá ser único en el documento'. Es un campo obligatorio.

    - **Medida externa de calidad.** Se corresponde con la calificación que está asociada al proyecto. Se trata de un valor numérico con decimales sin restricción de intervalos y es relevante para los proyectos de entrenamiento y no es utilizado en los proyectos gestionados por la aplicación. Será la cuarta columna de la hoja de cálculo y es un campo opcional.

- En la segunda hoja del documento de Excel, que se corresponde con la especificación de requisitos, existirán un conjunto de celdas que deberán ser cubiertas con el fin de rellenar los campos necesario:

    - **Identificador del proyecto.** Se trata del identificador del proyecto al cual pertenece el requisito. Es la primera columna de la hoja de requisitos y se trata de un campo obligatorio donde el valor debe ser el mismo que el especificado en la columna *Identificador* en la hoja correspondiente a la especificación de proyectos.

    - **Identificador del requisito.** Se trata del identificador que tendrá el requisito. Este campo sólo es respetado en caso de que se trate de un proyecto de entrenamiento, en caso de ser un proyecto gestionado por la aplicación se reemplazarán por un índice puramente numérico. Se trata de la segunda columna.

- **Título.** Se trata del título o la descripción breve que tendrá el requisito. Este campo es opcional y puede no ser rellenado. Se trata de la tercera columna de la hoja de requisitos.

- **Descripción.** Se trata de la descripción detallada que tendrá el requisito. Este campo es obligatorio y debe ser rellenado. Se trata de la cuarta columna de la hoja de requisitos.

A continuación, se muestra un ejemplo mediante ilustraciones en la que se ve la disposición que debe seguir la información en el documento:

A	B	C	D
Identificador	**Grupo**	**Nombre**	**Calidad**
1	2008/2009	Proyecto Road Island	9,5
2	2008/2009	Gestión de indicendias en las vías públicas	10

Ilustración 19: Ejemplo de la definición de un proyecto mediante un documento Excel

A	B	C	
Id. del Proyecto	**Identificador**	**Título**	**Descripción**
1	SRD-RF-01	Formato de tipo o tipos de una incidencia	El tipo de una incidencia será una lista de cadenas d
1	SRD-RF-02	Formato de gravedad de una incidencia	La gravedad de una incidencia será un número ente
1	SRD-RF-03	Formato de carretera implicada en una incidencia	La carretera implicada será una cadena de caracteres
1	SRD-RF-04	Formato de kilómetro de carretera implicada	El kilómetro de carretera implicada será un número

Ilustración 20: Ejemplo de la definición de requisitos mediante un documento Excel

II.1.3.3 Formato Microsoft Word 2003 y 2007

Otra de las formas en las cuales se pueden importar nuevos proyectos con el fin de que sean utilizados dentro del sistema es a través de un documento generado por la aplicación ofimática *Microsoft Word*, tanto en el formato de las versiones de 97 a 2003 (.doc) como en la versión 2007 y posteriores (.docx). Se permitirá de este modo la importación tanto de proyectos gestionados por la aplicación como proyectos de entrenamiento. Nuevamente y, al igual que ocurría en casos anteriores, deben especificarse un conjunto de normas de obligado cumplimiento para que el fichero pueda ser importado por el sistema:

- El documento de Word puede tener cualquier tipo de información asociada, sólo deberán etiquetarse las partes que sean relevantes para la identificación de requisitos. A partir de cada documento de Word se podrá importar únicamente un fichero.

- No se especificará ninguna información relevante respecto al proyecto, es decir, se asignará un nombre genérico al proyecto, el cual podrá ser cambiado una vez importado en la aplicación.

- Se utilizarán un conjunto de etiquetas similares a las utilizadas en lenguajes de marcado como XML o HTML. Se deberán utilizar las siguientes etiquetas:

  - <id>…</id>: Identifica únicamente un requisito en el proyecto

  - <title>…</title>: Engloba el título o descripción breve de un requisito contenido dentro del proyecto. Se trata de una etiqueta opcional.

- <description>…</description>: Encapsula la descripción detallada de un requisito contenido dentro del proyecto.

- Deben respetarse el orden de las etiquetas obligatorias. Es decir, debe haber una <id> y una <description> siempre en ese orden.

A continuación se muestra un ejemplo de cómo deberá ser etiquetado un documento con el fin de que sea correctamente importado por la aplicación:

Id: <id>SRD-RF-52</id>		Fuente: URD-RC-24
Título: <title>Modificación de los parámetros de backup</title>		
Necesidad: Conveniente	Prioridad: Media	Estabilidad: Variable
Descripción: <description>El sistema debe permitir mediante una operación a los usuarios con rol de Supervisor o Administrador modificar los parámetros del backup del Historial, pudiendo modificar la periodicidad, el día y la hora de realización.</description>		

Ilustración 21: Ejemplo de requisito en Word etiquetado para su procesamiento

II.1.3.4 Formato Microsoft Access 2003 y 2007

Otra de las formas de importar un proyecto a la aplicación *ReqStudio Plus* es a través de una base de datos realizada a través del software *Microsoft Access* dentro del paquete *Microsoft Office* en su formato de las versiones 97-2003 (mdb).Se permitirá de este modo la importación tanto de proyectos gestionados por la aplicación como proyectos de entrenamiento.

A través de este tipo de documento pueden definirse un número arbitrario de proyectos por documento, es decir, no es necesario un único documento por proyecto. Deberá utilizarse el modelo de datos que se muestra la Ilustración 22, cualquier dato adicional que se proporcione será ignorado por la aplicación:

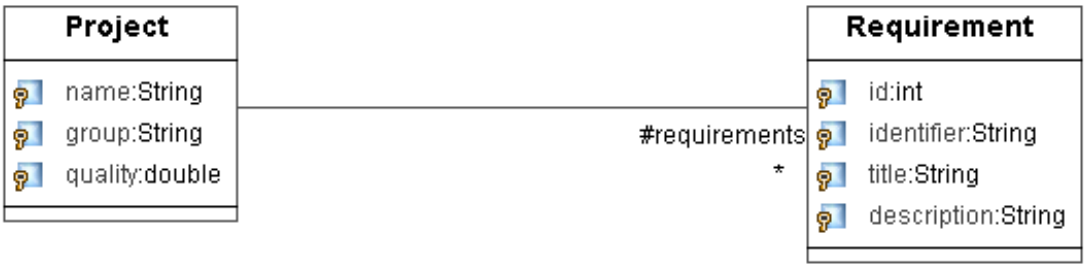

Ilustración 22: Modelo conceptual del formato de importación a través de un fichero Access

Los diferentes conceptos mostrados en el diagrama conceptual tienen el significado que se muestra en la siguiente tabla:

Concepto	Explicación
Project	Representa un proyecto que será importado en el sistema
Name	Representa el nombre único que deber tener el proyecto obligatoriamente
Group	Representa el grupo al cual pertenece el proyecto. Si no se asigna ningún valor, se asigna el valor "Ninguno"
Quality	Representa el valor de calidad del proyecto. Este valor sólo tiene relevancia en caso de que sea un proyecto de entrenamiento.
Requirement	Representa un requisito que pertenece al proyecto en el sistema.
Id	Representa el identificador único del requisito
Identifier	Representa el identificador del requisito dentro del proyecto
Title	Representa el título del requisito
Description	Representa la descripción del requisito

Tabla 32: Conceptos asociados al modelo de datos

II.1.3.5 Formato Fichero de texto plano

Otra de las formas de importar un proyecto a la aplicación ReqStudio Plus es a través de un fichero de texto plano. A partir del fichero de texto plano se pueden definir un conjunto arbitrario de proyectos. Es decir, no es necesario especificar para cada proyecto que se quiera importar un único fichero de texto. Al igual que en el resto de formatos, se deberán respetar un conjunto de normas:

- Se utilizará el mismo documento para la especificación tanto de proyectos como de requisitos.
- Se utilizarán campos separados por ";" similar a lo que se utiliza en los documentos CSV utilizando el punto y coma como separador.
- Los requisitos deberán estar situados inmediatamente debajo de la definición del proyecto. Es decir, no es posible definir primero los proyectos y luego los requisitos asociados a cada uno de ellos, tal y como se muestra en la siguiente imagen:

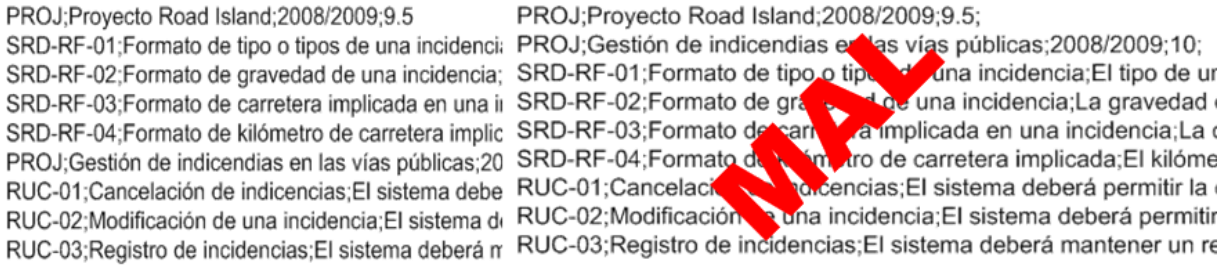

Ilustración 23: Muestra un ejemplo de la correcta organización así como una organización incorrecta de requisitos en un fichero de texto plano

- Para la definición de un proyecto deberá seguirse el siguiente patrón:
  - PROJ;<NAME>;<GROUP>;<QUALITY>
  - **PROJ.** Palabra clave que identifica un nuevo proyecto.

- **Name.** Nombre del proyecto
- **Group.** Grupo del proyecto, se trata de un campo opcional
- **Quality.** Medida externa de calidad del proyecto, se trata de un campo opcional.

- Para la definición de un requisito deberá seguir el siguiente patrón:
  - <ID>;<TITLE>;<DESCRIPTION>

A continuación se muestra un ejemplo, de un fichero de texto, que sería aceptado por parte del sistema para su correcta importación:

```
1  PROJ;Proyecto Road Island;2008/2009;9.5
2  SRD-RF-01;Formato de tipo o tipos de una incidencia;El tipo de una incidencia
3  SRD-RF-02;Formato de gravedad de una incidencia;La gravedad de una incidencia
4  SRD-RF-03;Formato de carretera implicada en una incidencia;La carretera impli
5  SRD-RF-04;Formato de kilómetro de carretera implicada;El kilómetro de carrete
6  PROJ;Gestión de indicendias en las vías públicas;2008/2009;10;
```

Ilustración 24: Muestro del ejemplo de un proyecto importable en texto plano

II.1.4 Resumen de la importación/exportación de proyectos

Una vez explicados los diferentes formatos de importación y exportación de cada uno de los formatos soportados por la aplicación, a continuación se muestran una tabla explicativa que resume lo anteriormente comentado:

Formato	Proyecto completo	Requisitos individuales	Proyecto gestionado	Proyecto de entrenamiento
Requirements Studio Plus	Sí	Sí	Sí	Sí
Requirements Studio 2.1	Sí	Sí	Sí	No
Fichero EXCEL	Sí	Sí	Sí	Sí
Fichero WORD	Sí	Sí	Sí	Sí
Fichero ACCESS	Sí	Sí	Sí	Sí
Fichero de texto plano	Sí	Sí	Sí	Sí

II.2 Importación/Exportación de otras entidades.

Además de permitir la importación y exportación de proyecto tanto gestionados por la aplicación como de entrenamiento, existen otras entidades cuya importación y exportación se considera necesaria. A pesar de que sea posible, la compartición del acceso a ciertas entidades, se considera útil una exportación en un fichero en almacenamiento secundario con el fin de poder importarlos en otras instalaciones diferentes de la aplicación.

Todas las entidades exportables/importables tendrán únicamente soporte mediante la importación/exportación mediante XML a través del *framework XMLSerialization* propio de .NET. Las entidades exportables/importables son:

- **Dominio.** En el caso de la exportación e importación, se realiza solo de los términos asociados al dominio y no de los usuarios asociados al mismo. Nuevamente, al igual que ocurría con los proyectos, se supondrá que el único usuario inicialmente es el que realiza la importación, el cual tendrá todos los permisos. Su fecha de última modificación será en la que se realice la importación.

- **Lista de términos especiales.** Ocurre lo mismo que el dominio, se importan y exportan los términos asociados pero no los permisos de los usuarios.

- **Métricas.** Se permite la exportación de los diferentes indicadores y sus valores asociados, así como los valores agregados y nominales para los diferentes valores de calidad. Nuevamente, no se exportan los usuarios.

- **Experimentos.** Se permite la exportación de un experimento en el cual se exportarán los datos de afinamiento del algoritmo genético, los valores de calidad tanto nominales como agregados, el conjunto de entrenamiento utilizado compuesto por los proyectos seleccionados con sus requisitos asociados, la población inicial y los recuentos utilizados en el experimento. Nuevamente, no se exportan los permisos de los usuarios con acceso al mismo.

Anexo III. Recuentos establecidos por defecto

ID	Nombre	Tipo	Categoría	Recuento CAKEIndexer
1	Tamaño medido en caracteres	Convexo	Morfológico	TextLength
2	Tamaño medido en sílabas	Convexo	Morfológico	SyllableCount
3	Tamaño medido en palabras	Convexo	Morfológico	WordCount
4	Tamaño medido en frases	Convexo	Morfológico	PhraseCount
5	Tamaño medido en párrafos	Convexo	Morfológico	ParagraphCount
6	Legibilidad de Flesch-Kinkaid	Decreciente	Morfológico	Legibility
7	Caracteres entre signos de puntuación[4]	Convexo	Morfológico	CharsBetweenPunctuationMarks
8	Acrónimos y abreviaturas	Decreciente	Morfológico	AcronymnCount
9	Formas verbales imperativas	Convexo	Analítico	ImperativeVerbCount
10	Formas verbales condicionales	Decreciente	Analítico	ConditionalVerb
11	Sustantivos del dominio	Convexo	Analítico	DomainNounCount
12	Verbos del dominio	Convexo	Analítico	DomainVerbCount
13	Sustantivos totales	Convexo	Analítico	NounCount
14	Verbos totales	Convexo	Analítico	VerbCount
15	Términos totales	Convexo	Morfológico	TermCount
16	Tokens totales	Convexo	Morfológico	TokenCount

[4] Este recuento proporcionado por CAKE Indexer calcula el promedio de caracteres entre signos de puntuación, es decir, el cociente (longitud del texto en caracteres / signos de puntuación). Este cociente es el inverso de la medida propuesta en el aparatado 2.1.2.3, "número de signos de puntuación por frase, dividido por la longitud de la frase" (signos de puntuación / longitud del texto en caracteres). Ambas medidas son convexas (los valores grandes pasan a ser pequeños y viceversa, los valores intermedios siguen siendo los preferibles).